VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Ovládání technologie mobilním zařízením

The technology control by mobile device

2012 **Bc. Radek Podešva**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Radek Podešva**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Ovládání technologie mobilním zařízením
The Technology Control by Mobile Device

Zásady pro vypracování:

Výsledkem práce bude možnost řídit technologii ovládanou PLC Simatic S7 300 prostřednictvím mobilního zařízení.

1. Prostudovat možnosti komunikace komunikačního procesoru CP343-IT za účelem propojení PLC a mobilního zařízení, zda jít cestou přes externí server nebo přes přímé zápisy do paměti CP343-IT.
2. Nastudovat a naprogramovat FTP spojení s vnějším světem na externí server (v případě varianty komunikace přes externí server).
3. Nastudovat možnosti komunikačního procesoru CP343-IT za účelem propojení PLC a mobilního zařízení.
4. Nastudovat a popsat komunikační protokoly SIMATIC S7 Ethernet Fetch/Write a SINEC - H1.
5. Vypracovat aplikaci, která bude obsahovat sekci nastavení vlastního hardware.
6. Vytvořit aplikaci pracující na operačním systému Symbian nebo Windows Mobile pomocí programovacího jazyka Java nebo C# umožňující pomocí jednoduchého user-friendly interface zadávat technologické povely a data.
7. Předvedení konkrétní aplikace pro řízení ohřevu užitkové vody, topné vody a topení v domě.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Josef Podešva**

Konzultant diplomové práce: Ing. Michal Krumnikl

Datum zadání: 19.11.2010

Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * FAKULTA elektrotechniky a informatiky * OSTRAVA *

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: *04.05.2012*

Poděkování

Rád bych poděkoval Ing. Michalovi Krumniklovi Ph.D: za odbornou pomoc a konzultaci při vytváření této diplomové práce, svému vedoucímu diplomové práce Ing. Josefu Podešvovi za spousty přínosných poznámek při práci s řídicími systémy a své rodině za podporu.

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.“

Dne: *04.05.2012*

Podpis zástupce

Abstrakt

Tato diplomová práce se zabývá studiem možností propojení PLC s mobilními aplikacemi za účelem monitorování a zadávání technologických povelů. Popisuje komunikační procesor CP343-1 IT pro PLC SIMATIC S7-300, jeho komunikační možnosti a komunikační protokoly. V další části pak pojednává o realizaci spojení PLC s mobilní aplikací využitím komunikační knihovny LibNoDave, jež je implementována na platformě Google Android.

Klíčová slova

PLC, SIMATIC, CP343-1 IT, RM ISO/OSI, TCP, ISO-on-TCP, libnodave, JAVA, Android

Abstract

This diploma thesis deals with a study of connection possibilities between PLC and mobile application for monitoring and technological commands sending reasons. It describes the communication processor CP343-1 IT for PLC SIMATIC S7-300, its communication possibilities and communication protocols. In the next section the thesis discusses the realization of the connection between PLC and mobile application using communication library LibNoDave, which is implemented on the Google Android platform.

Key words

PLC, SIMATIC, CP343-1 IT, RM ISO/OSI, TCP, ISO-on-TCP, libnodave, JAVA, Android

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
PLC	Programmable Logical Controller	Programovatelný logický automat
CPU	Central Processing Unit	Centrální procesorová jednotka
CP	Communication Processor	Komunikační procesor
DI	Digital Input	Digitální vstup
DO	Digital Output	Digitální výstup
AI	Analog Input	Analogový vstup
AO	Analog Output	Analogový výstup
DB	Data Block	Datový blok
M	Merker	
I	Input	
OSI	Open System Interconnection	Otevřený systém propojování
FTP	File Transfer Protocol	Protokol pro přenos souborů
UDP	User Datagram Protocol	
TCP	Transmission Control Protocol	
IP	Internet Protocol	Internetový protokol
IDE	Integrated Development Environment	Integrované vývojové prostředí
SDK	Software Development Kit	Softwarový vývojový balíček

Obsah

1 Úvod

1.1 Cíl práce

Prvořadým cílem této diplomové práce je vytvořit aplikaci pro mobilní zařízení, která bude schopna komunikovat s PLC skrze jednoduché uživatelsky přívětivé rozhraní. Tato mobilní aplikace bude vytvořena na zakázku dle požadavků firmy SIMPO CZ, s.r.o., která se zabývá automatizací řídících systémů SIMATIC firmy Siemens AG. Nápad na tuto zakázku vznikl kvůli rychlému rozvoji mobilních technologií a chytrých telefonů, a také protože jsem s firmou spolupracoval již v minulosti a seznámil jsem se u ní s řídícími systémy. Možnost ovládat technologický proces kdekoli a kdykoli skrze mobilní telefon se pak jevila jako výborný námět na téma diplomové práce. Firma souhlasila s nápadem a nabídla možnost vypracování diplomové práce na toto téma.

Výsledkem by tedy měla být mobilní aplikace implementována v jazyce JAVA pro platformu Google Android, která by měla komunikovat s PLC umístěném v rodinném domě.

1.2 Motivace

Zásadní motivací při výběru tématu bylo nadšení z chytrého mobilního telefonu HTC HD2, na který jsem si nainstaloval operační systém Google Android. Tento operační systém byl zpočátku velice neobvyklý pro běžného uživatele operačního Windows Mobile, ale rychle jsem si zvykl na nové intuitivní a jednoduché ovládání a začal se zajímat o vývoj aplikací na platformě Google Android. Díky internetu dnes máme přístup k různým vývojářským prostředím, které usnadňují psaní aplikací pro mobilní telefony. Od roku 2005, kdy společnost Google koupila malou kalifornskou firmu Android Inc., se vývoj aplikací posunul daleko kupředu. Google Market (místo, odkud se dají kupovat a stahovat aplikace pro operační systém Google Android) se závratnou rychlostí začal zaplňovat různými aplikacemi. Vývoj aplikací pro tento operační systém je podle mého názoru velice snadný a zábavný, proto si získal přízeň mnoha začínajících vývojářů.

Protože jsem spolupracoval s firmou SIMPO CZ, s.r.o. již dříve v rámci bakalářské práce, měl jsem možnost seznámit se s průmyslovým prostředím a s řídícími systémy. Pracoval jsem s PLC SIMATIC S7-300 a už tehdy mne zajímala možnost jiného způsobu ovládání technologie, než pouze přes lokální dotykové ovládací panely, případně přes vizualizační stanice, což byly vlastně stolní nebo průmyslové počítače se SCADA systémy. Nabízela se myšlenka ještě další, a to přes zadávání parametrů nebo čtení dat technologie řízené řídícím systémem Simatic prostřednictvím mobilní technologie.

2 Obecné informace

V této kapitole se nachází vysvětlení základních pojmů, které tvoří informační pozadí strany řídícího systému.

2.1 Automatizace

Automatizace znamená v jednodušším pojetí využití řídících systémů a informačních technologií ke snížení potřeby lidského zásahu do technologického procesu. V průmyslovém prostředí je automatizace o krok před mechanizací, protože zavedením mechanizace do technologického procesu se práce člověku usnadní, kdežto zavedením automatizace se daná práce vykoná bez, anebo s minimálním zásahem člověka. V současnosti hraje automatizace stále důležitější roli jak ve světové ekonomice, tak i v běžném životě.

Automatizace se původně začala rozšiřovat nejvíce v průmyslové výrobě, na kterou měla obrovský vliv. Automatizované linky eliminovaly přítomnost lidské pracovní síly a tím snížily náklady na provoz. Podobně se automatizace rozvinula i v telekomunikacích, kdy se místo lidských operátorů zavedly automatizované telefonní ústředny, což taktéž ušetřilo spousty peněz svým provozovatelům. V medicíně automatizace pomohla urychlit a upřesnit laboratorní analýzy lidských genů, buněk, krve a vláknin. Bankomaty jsou také automatizované stroje zkracující fronty zákazníků v bankách. Automatizace je zkrátka jedním z nejdůležitějších pokroků minulého století, který má za následek posun světové ekonomiky kupředu.

Automatizace s sebou přináší různé výhody i nevýhody.

Výhody:

- Dá se využít pro vykonání monotónní a pro člověka fyzicky náročné práce.
- Může nahradit člověka v prostředí, které je pro člověka nebezpečné (např.: prostor s nebezpečím ozáření radiací, vesmír, oheň, mořské dno...)
- Automatizované stroje mohou nahradit člověka v případech, kdy jsou požadované určité fyzické vlastnosti, kterých není člověk schopen dosáhnout (např.: velikost, hmotnost, rychlost, odolnost, síla...).
- Značně snižuje časovou náročnost pro vykonání určité práce.
- Využitím automatizace se mohou uvolnění pracovníci zabývat jinou pracovní činností.
- Poskytuje pracovní pozice pro vývojáře, údržbáře a techniky automatizovaných procesů.

Nevýhody:

- Automatizace je v dnešní době stále ještě drahá záležitost. Cena zavedení automatizace může v některých případech převýšit částku, kterou měla automatizace původně ušetřit.

V průběhu času vznikaly na automatizaci různé požadavky na zavedení větší flexibility a přenositelnosti. Když měla být například automatizovaná linka na výrobu produktu A zrušená a nahrazená automatizovanou linkou na výrobu produktu B, nejjednodušší by bylo tuto automatizaci jednoduše "přenést" ze staré linky na novou. S příchodem digitální techniky se začaly v automatizaci používat digitální prvky, které byly v mnoha případech daleko spolehlivější a přesnější, než jejich analogové ekvivalenty. Tento pokrok pak nabídl lepší konfigurační, parametrizační a operační možnosti. Současně se tak s příchodem rodiny komunikačních protokolů Fieldbus usnadnila realizace automatizace v takovém rozsahu, aby uspokojila poptávku po již zmiňované flexibilitě a přenositelnosti.

Automatizovaný technologický proces je řízen automatizačním nástrojem. Nejznámějším z nich je programovatelný logický automat (PLC).

2.2 PLC

PLC neboli Programmable Logical Controller je menší počítač používaný v automatizaci technologických procesů. Je často hojně využívaný v továrnách na montážních linkách, v zábavních parcích, nebo také pro vánoční osvětlení domu. Oproti řízení technologického procesu počítačem má PLC možnost zpracování mnohonásobých vstupů a výstupů, je odolný vůči většímu rozsahu teploty, je imunní vůči elektrickému rušení a úspěšně vzdoruje i vibracím a pádu. Řídící program je většinou uložen v nevolatilní energeticky zajištěné paměti. Tento program se u PLC vykonává v tzv. cyklech.

Většina PLC obsahuje následující komponenty:

- Centrální procesorová jednotka (CPU)
    - Zpracovává program uložený v paměti PLC.
- Digitální vstupy (DI), Digitální výstupy (DO)
    - Přijímají a vysílají digitální signály.
- Analogové vstupy (AI), Analogové výstupy (AO)
    - Přijímají a vysílají analogové signály.

- Komunikační procesor (CP)
    - Zprostředkovává komunikaci PLC s okolím, umožňuje sběr a přenos dat po specifickém komunikačním rozhraní.
- Zdroj

Z pohledu konstrukce se PLC dělí na následující skupiny:

- Kompaktní
- Modulární

Kompaktní systém PLC obsahuje výše zmíněné komponenty v jednom modulu a jeho možnost rozšíření o další komponenty je velice omezená. Naproti tomu modulární systém PLC se skládá z dílčích modulů vzájemně propojených do jednotného celku a možnost rozšíření není nijak limitovaná.

Příklady kompaktního PLC systému jsou *VersaMax Micro/Micro 64* firmy GE Intelligent Platforms, nebo *CP1H* firmy OMRON.

Příklady modulárního PLC systému jsou *Micro800 System* značky Allan-Bredley firmy Rockwell Automation, nebo právě *SIMATIC S7-300* firmy Siemens AG.

2.3 Řídicí systémy Simatic

V evropském měřítku jsou v automatizaci technologických prostředků nejvíce nasazovány řídicí systémy Simatic. Řídicí systémy SIMATIC jsou známy především svojí spolehlivostí a robustností. Již řadu let jsou stabilním prvkem nejrůznějších technologií. Své renomé si získala dnes už výběhová řada SIMATIC S5, která na mnoha místech v České republice ještě spolehlivě slouží již déle než 20 let. Na ni úspěšně navázala řada SIMATIC S7, která pokračuje v tradici řídicích systémů firmy Siemens, naprosto spolehlivých a využitelných ve všech oblastech průmyslové automatizace.

Řídicí systémy Simatic se dělí na:

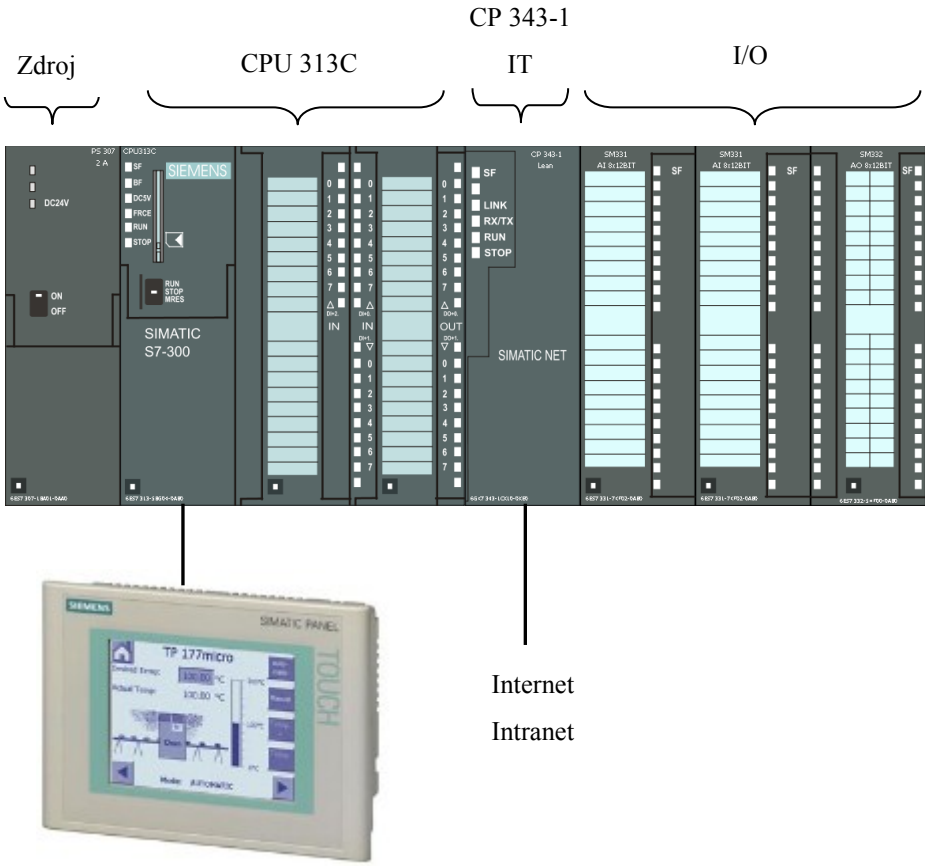
- Mikrosystémy (LOGO!, SIMATIC S7-200 a SIMATIC S7-1200) – malé kompaktní i modulární řídicí systémy s nabídkou nepříliš početných vstupů/výstupů, cenově velmi příznivé a programovatelné pomocí jednoduchého programovacího jazyka
- Systémy C7 (řídicí systém na bázi S7300 a panel v jednom přístroji)
- Řídicí systémy řady S7 300/400 – špička v nabídce firmy Siemens, obsahující obrovské spektrum možností jak modulárních, tak programování. (a současně nejrozšířenější z hlediska nasazení)

V této diplomové práci budu využívat a spolupracovat s řídicím systémem Simatic S7 300. Jedná se o standardní jednotku vybavenou standardními moduly s unifikovanými signály (24V logika

I/O), řídící technologii ohřevu topné a užitkové vody v rodinném domě. Zdrojem tepla jsou solární kolektory a plynový kotel, využití tepla je určeno pro ohřev užitkové vody, topení a ohřev bazénu. Jako prostředník pro ovládání je součástí konfigurace řídícího systému lokální dotykový panel, pro komunikaci s vnějším prostředím je zde komunikační karta, která slouží pro vzdálenou správu řídícího systému (programování, diagnostika, parametry technologie) a současně i jako web server, který obsahuje jednoduchou aplikaci pro zobrazení dat a ovládání základních parametrů technologie prostřednictvím internetu nebo intranetu.

2.3.1 Konkrétní konfigurace PLC

Na níže uvedeném obrázku je zobrazena konkrétní konfigurace řízení ohřevu.



Obrázek 2.1 – Příklad konfigurace řídícího systému SIMATIC

Význam jednotlivých zkratek:

CPU 313C – procesor 313 compact – součástí kompaktu je i jednotka vstupů a výstupů (24DI/16DO/5AI/2AO)

CP 343-1 IT – komunikační procesor, obsahující SEND/RECEIVE a FETCH/WRITE interface, umožňující UDP, TCP, S7 komunikaci, routing, web server a e-mail, NTP, FTP, IP konfiguraci DHCP nebo programově (HW konfigurací nebo na úrovní aplikačního SW Step 7)

I/O - moduly vstupů/výstupů

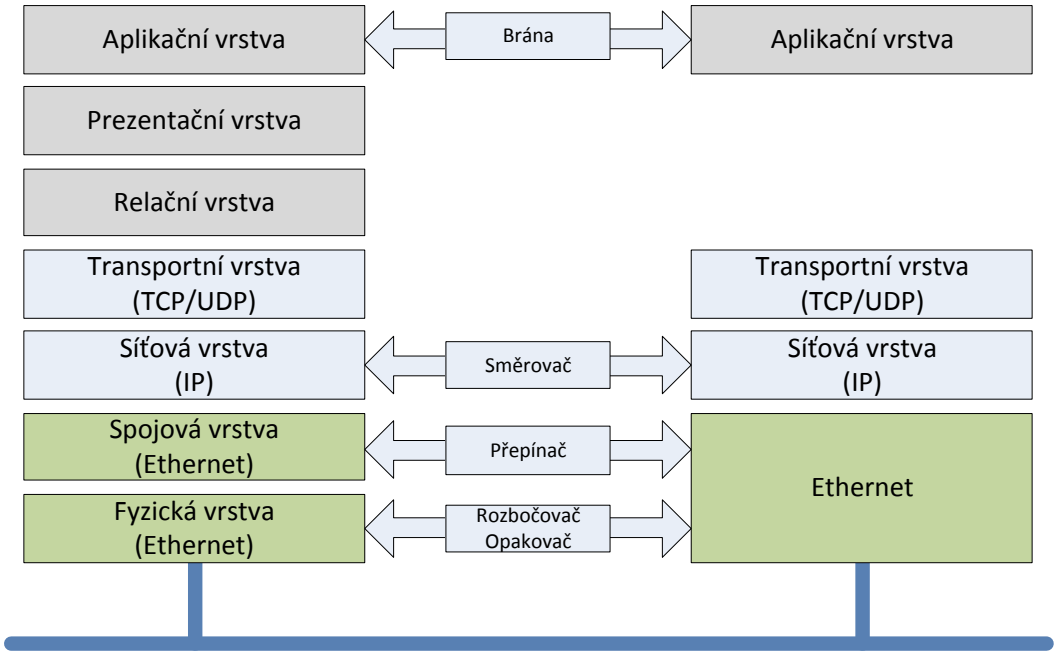
2.4 CP343-1 IT

Komunikační procesor CP343-1 IT je určen pro komunikaci na síti Ethernet, kde je schopen komunikovat s partnery na různých vrstvách tzv. ISO/OSI reference modelu, který je standardem pro Ethernet komunikaci. Prostředkem komunikace s okolním světem je komunikační procesor, jehož zkrácené a používané označení je CP 343-1 IT [8]. Tento procesor poskytuje:

- UDP, TCP, S7 komunikaci
- SEND/RECEIVE a FETCH/WRITE rozhraní
- Routing
- web server a e-mail
- NTP, FTP
- IP konfiguraci pomocí DHCP nebo programově (HW konfigurací nebo na úrovní aplikačního SW Step 7)

2.4.1 Komunikační struktura dle ISO/OSI reference modelu

Nejprve by bylo vhodné popsat referenční model OSI dle ISO normy, o který se většina komunikačních protokolů opírá. Tento referenční model je složen ze sedmi základních vrstev, které mezi sebou vzájemně komunikují. Komunikace však probíhá pouze mezi sousedními vrstvami. Jednotlivé vrstvy tedy vypadají následovně:



Obrázek 2.2 – Referenční model ISO/OSI

1) Fyzická vrstva

Fyzická vrstva je zodpovědná za předávání dílčích informací na spojovací médium a přijímání ze spojovacího média. Tato vrstva se nezabývá obsahem informace, kterou zpracovává, ale elektrickými a mechanickými vlastnostmi signálů a signálními metodami.

2) Spojová vrstva

   Tato vrstva se stará o úplnost přenosu dat, jestli nebyla data při přenosu ztracena nebo změněna. Přenášené bity jsou rozděleny do rámců, které mají jasně definovanou strukturu a obsahují informace potřebné k validaci úspěšného přenosu.

3) Síťová vrstva

   Síťová vrstva vytváří na trase mezi vysílačem a přijímačem seznam síťových bodů, obvykle směrovačů. Nejznámějším protokolem třetí vrstvy je IP protokol TCP/IP.

4) Transportní vrstva

   Tato vrstva je zodpovědná za úplný přenos paketů z jednoho konce na druhý a stará se o integritu paketu. Nižší vrstvy mohou zahazovat pakety, ale transportní vrstva provádí sekvenční kontrolu dat.

5) Relační vrstva

   Tato vrstva se stará o upořádanou komunikaci a koordinaci mezi dvěma body.

6) Prezentační vrstva

   Tato vrstva umožňuje překlad přenosu mezi různými systémy.

7) Aplikační vrstva

   Tato horní vrstva definuje jazyk a syntaxe, které programy používají ke komunikaci s jinými programy. Aplikační vrstva představuje účel komunikace na prvním místě.

Bližší informace o RM OSI na [1].

2.4.2 Industrial Ethernet (dříve SINEC H1)

Industrial Ethernet je následníkem Ethernet komunikace SINEC H1. Poskytuje v současné době nejrychlejší komunikaci v automatizaci a je následován průmyslovou sítí PROFINET (obdobně jako PROFIBUS, PROFINET umožňuje propojení mezi řídícími systémy a akčními členy polní instrumentace s rozhraním pro PROFINET).

SINEC H1 je tedy dřívější označení pro Industrial Ethernet, který používá dvě vrstvy ISO/OSI reference modelu (č.4 – Peer Services a č.7 – TF services), pro komunikaci mezi dvěma partnery. Využívá ISO Transport typ komunikace.

Peer Services	**TF services**
Až 64 Kb dat pro single job	Až 31 TF aplikací
Až 40 transport connections	Přístup na proměnné (čtení/zápis)
Komunikace s S5 PLC	Pojmenované proměnné (např. „P-Čerpadlo“)
Čtení peer data transfer	Kódované proměnné (např. P1001)
Zápis peer data transfer	Čtení/Zápis strukturovaných proměnných (polí proměnných)
Send/Receive peer data transfer	

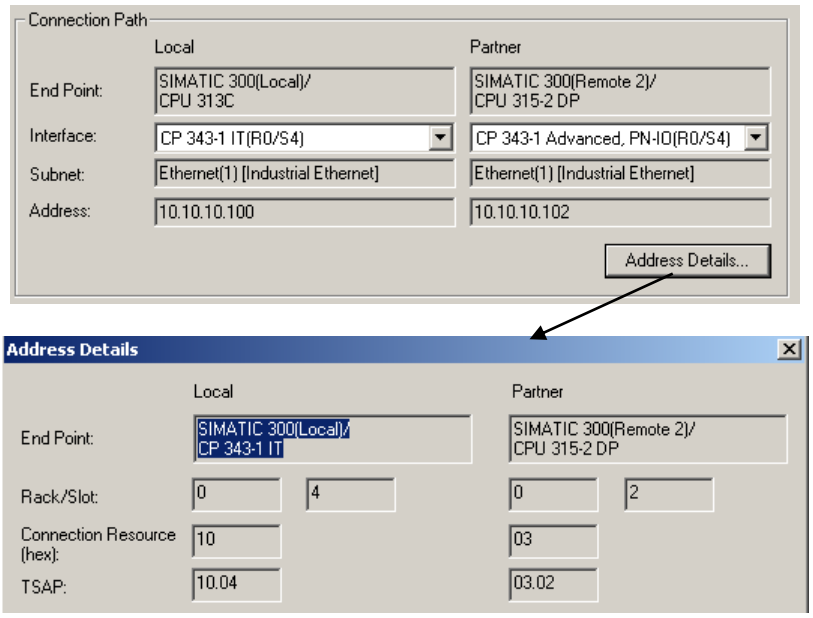
Bližší informace se dají nalézt na [4].

2.4.3 Typy komunikace

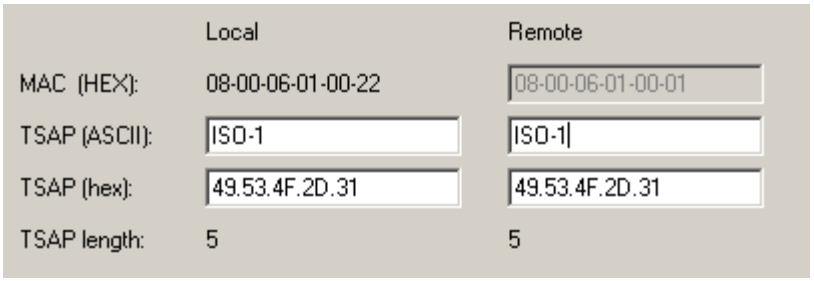
CP343-1 IT nám umožňuje komunikaci následnými protokoly:

- **S7 komunikace** – tato komunikace je určena především pro Simatic S7 partnery. Základními parametry komunikace mezi lokálním a vzdáleným partnerem jsou:
  - IP adresa
  - Číslo **racku,** kde se nachází CPU/CP
  - Číslo **slotu**, kde se nachází CPU/CP

Kombinace čísel Rack/Slot pak je součástí TSAP parametru (Transport Service Access Point). Potvrzení přenosu dat probíhá prostřednictvím vrstvy č.7 ISO/OSI reference modelu.



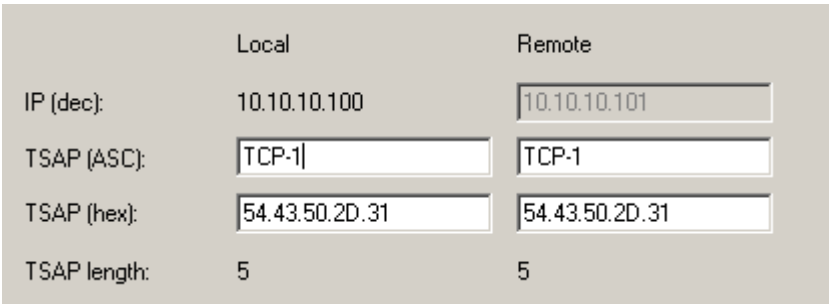
- **ISO Transport komunikace** – tato komunikace je starším typem komunikace, která byla využívána především u komunikačních partnerů Simatic S5 nebo PC na bázi ISO Transport. ISO Transport využívá vrstvu 4 ISO reference modelu. Pro přenos dat lze použít Send/Receive nebo Fetch/Write utility. Základními parametry komunikace mezi lokálním a vzdáleným partnerem jsou:
    - MAC adresa
    - Kombinace znaků a číslic určující TSAP (Transport Service Access Point)



- **ISO on TCP Transport komunikace** – tato komunikace umožňuje komunikovat na TCP/IP standardu s využitím RFC1006 na vrstvě 4 ISO reference modelu. RFC 1006 určuje, jak budou služby ISO vrstvy 4 namapovány na TCP. Pro přenos dat lze použít Send/Receive nebo Fetch/Write utility.

Základními parametry komunikace mezi lokálním a vzdáleným partnerem jsou:

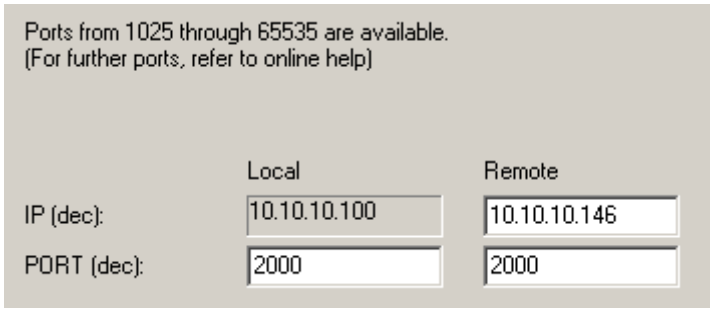
- IP adresa
- Kombinace znaků a číslic určující TSAP (Transport Service Access Point)



- **TCP komunikace** – splňuje TCP/IP standard. Umožňuje komunikovat s partnery, kteří splňují využití TCP/IP standardu (PC, non Siemens zařízení). Pro přenos dat lze použít Send/Receive nebo Fetch/Write utility.

Základními parametry komunikace mezi lokálním a vzdáleným partnerem jsou:

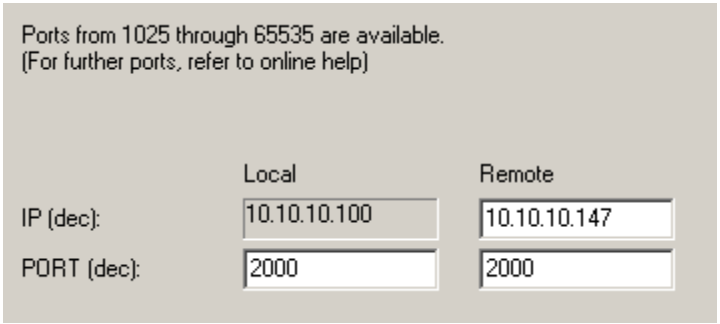
  - IP adresa
  - Port



- **UDP komunikace** – splňuje TCP/IP standard. Umožňuje komunikovat s partnery, kteří splňují využití TCP/IP standardu (PC, non Siemens zařízení). Jedná se o nezabezpečený přenos dat. Pro přenos dat lze použít pouze Send/Receive utility.

Základními parametry komunikace mezi lokálním a vzdáleným partnerem jsou:

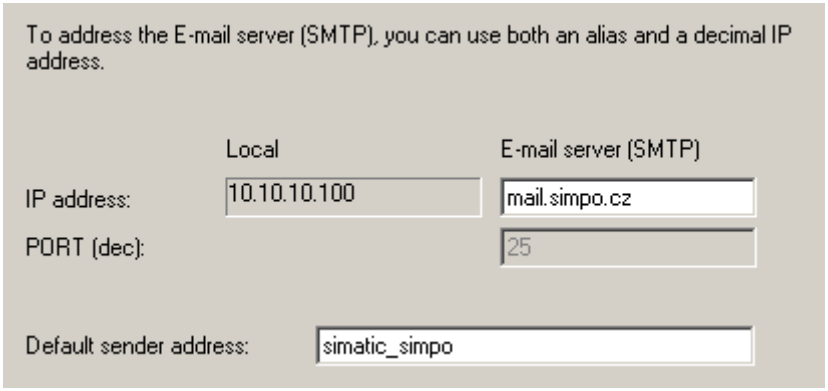
  - IP adresa
  - Port



- **E-Mail komunikace** – splňuje TCP/IP standard.

Základními parametry komunikace mezi lokálním a vzdáleným partnerem jsou:

- IP adresa
- Vzdálený SMTP server (definovaný IP adresou nebo doménovým jménem)
- Jméno odesílatele



2.4.4 Komunikační utility

Mezi komunikační utility patří Send/Receive nebo Fetch/Write utility.

SEND/RECEIVE

Tento interface patří mezi základní způsob komunikace mezi dvěma Siemens partnery řady Simatic S7.

FETCH/WRITE

Pokud zvolíme jako platformu komunikace ISO-on-TCP, můžeme přistupovat do systémové paměti Simatic S7 PLC přímo ze Simatic S5 nebo obecné „non Siemens“ stanice.

FETCH – čtení dat přímo ze systémové paměti

WRITE – zápis dat přímo do systémové paměti

Nastavení jednotlivých partnerů z hlediska řízení komunikace:

- SIMATIC S7 - PASSIVE connection
- PC stanice - Active connection

3 LibNoDave

V této kapitole si přiblížíme detailně, co to knihovna LibNoDave vlastně je, jak byla navrhnuta, jakým způsobem probíhá komunikace mezi PLC a aplikací pomocí této knihovny.

3.1 Popis knihovny

Knihovna LibNoDave umožňuje přenos dat mezi počítačem a řídícím systémem SIMATIC firmy Siemens AG. Knihovna je zdarma dostupná na adrese http://libnodave.sourceforge.net [3] a podléhá licenční politice na základě podmínek GNU LGPL – Všeobecná veřejná licence GNU pro knihovny [2], která je příkladem Copyleftové licence. Copyleft je opakem Copyrightu a poskytuje ochranu svobodného použití knihovny i její modifikace.

Knihovnu vytvořil německý programátor Thomas Hergenhahn za účelem nekomerční náhrady knihovny proDave firmy Siemens AG. Protože nebyla vydána žádná dokumentace ke komunikaci mezi počítačem a PLC SIMATIC, která by jednoznačně usnadnila vývoj takovéto knihovny, musel autor odchytávat datový tok a na tomto základě byl schopen knihovnu napsat. Původně ji napsal v Linuxu pro Linux v roce 2002. První verze měla označení libnodave-0.1, poslední verzí je libnodave-0.8.4.6 z roku 2011. Autor knihovnu pravidelně aktualizoval a rozšiřoval její možnosti, takže se od verze libnodave-0.5 můžeme setkat s funkčními variacemi pro různé programovací jazyky, jako jsou C++, C#, Delphi, Pascal, Perl, Visual Basic, i Visual Basic for Applications. Knihovna byla v roce 2005 také přepsána do jazyka JAVA, ale v tomto programovacím jazyce se nedočkala další aktualizace a zůstala ve verzi libnodave-java-0.1. Pro každý jazyk byly vytvořeny i funkční ukázky a součástí balíčku jsou i podpůrné testovací skripty.

LibNoDave představuje konkurenceschopný nástroj, který dokáže pracovat s daty v PLC stejně kvalitně, jako konkurenční knihovny zmíněné v kapitole ***Error! Reference source not found.*** *omunikační knihovny*. Je sice pravda, že tyto konkurenční knihovny nabízejí jistý komfort v podobě hotového produktu, u kterého nemusí uživatel vynaložit žádné větší úsilí, ale na druhou stranu věnováním času k vytvoření aplikace použitím knihovny LibNoDave může uživatel dosáhnout daleko vyšší míry spokojenosti díky odměně v podobě aplikace naimplementované na míru. Protože jsou mobilní aplikace pro Google Android vyvíjené v jazyce JAVA, budu dále popisovat možnosti knihovny pouze pro verzi libnodave-java-0.1.

Podporovaným přenosem dat knihovny je využitím ISO modelu TCP/IP protokolu skrze Ethernet. Komunikuje se s komunikačními procesory řady 243, 343 a 443 včetně jejich –IT verzí. Dále knihovna podporuje komunikační knihovny a ovladače firmy Siemens AG, které umožňují práci s daty

přes MPI adaptéry CP5511, CP5611 a USB-MPI, a zároveň i další MPI a PPI adaptéry skrze sériové rozhraní.

3.2 Třídy v libnodave-java-0.1

Balíček obsahuje několik tříd, které zastávají různé funkce v průběhu komunikace. Vzhledem k tomu, že k těmto třídám neexistuje dokumentace, pokusím se popsat třídy podle jejich funkčnosti. Zaměřím se na třídy, které jsou dále použity v samotném vypracování aplikace.

Seznam hlavních tříd:

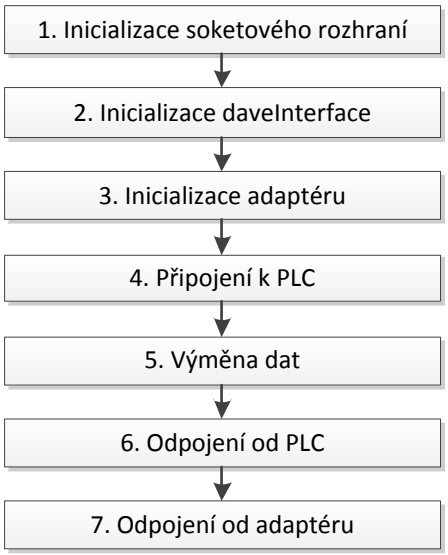
- Nodave
  - Tato třída obsahuje konstanty používané při komunikaci a potřebné metody pro manipulaci s datovým tokem bajtů.
- PDU
  - V této třídě dochází k enkapsulaci datového paketu posílaného do PLC. Sestavují se zde PDU pro čtení a zápis.
- S7Connection
  - Třída, která převede komunikaci z libovolného typu spojení do S7Connection komunikačního protokolu, kterému PLC rozumí. Obsahuje důležité metody pro čtení a zápis hodnot.
- PLCinterface
  - Stará se inicializaci MPI adaptéru a jeho odpojení.
- Result
- ResultSet
- S7DateTime

Seznam tříd specifických pro různé typy komunikace:

- TCPConnection
  - Tato třída zprostředkovává spojení skrze protokol TCP, jako parametry spojení používá IP adresu a port.
- TCP243Connection
- PPIConnection
- MPIConnection (včetně třídy MPIinterface)
- IBH_MPIConnection
- NetLinkProConnection (včetně třídy NetLinkProInterface)

3.3 Princip komunikace

Komunikace knihovny libnodave s PLC se dá popsat několika kroky. Nejlépe to vystihuje následující diagram:



Obrázek 3.1 – Postup komunikace knihovny libnodave

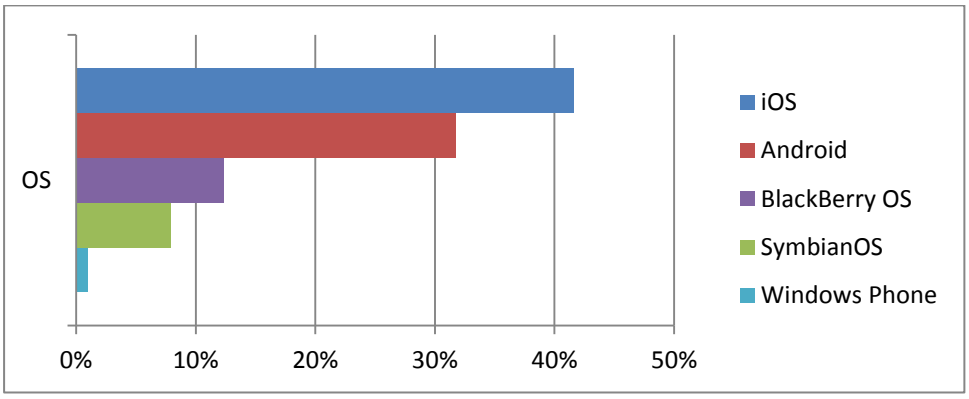
Principiálně knihovna funguje tak, že vytvoří spojení s PLC prostřednictvím komunikace na protokolu ISO-on-TCP. Toto spojení je realizováno v prvních třech krocích. Komunikace probíhá vyměňováním PDU, které knihovna libnodave na základě třídy Nodave umí enkapsulovat. Tyto PDU se posílají přímo do CPU PLC a obsahují sekvence bytů určující, zda se jedná o informaci pro přečtení hodnoty z paměťové oblasti v PLC, nebo zápis. Autor knihovny tyto bytové sekvence odchytil v komunikaci mezi PLC a programem Step7 a díky tomu byl schopen jasně deklarovat, co má jaká bytová sekvence na starost.

Jakmile dojde k výměně dat, zavolají se metody pro odpojení od PLC a ukončení spojení s adaptérem.

4 Analýza a návrh mobilní aplikace

Před samotným vypracováním aplikace bylo nutné provést malý průzkum trhu. V původním zadání se měla aplikace implementovat na platformě Windows Mobile, případně pro Symbian. Protože se rozvoj informačních a telekomunikačních technologií posouvá dopředu závratnou rychlostí, musel jsem přehodnotit situaci a jako platformu zvolit mladší operační systém Google Android.

Google Android je otevřená platforma nabízející neuvěřitelné možnosti, jak se vyžít jako programátor. Vývoj aplikací je relativně snadný, intuitivní, ale hlavně zábavný. Protože si získává Google tímto operačním systémem přízeň mnoha zákazníků, ve výsledku by se tak s rostoucím počtem chytrých telefonů podporujících tento operační systém dalo zasáhnout větší cílovou skupinu uživatelů, než například tu používající mobilní operační systém Symbian, případně Windows Phone. Podle serveru http://gs.statcounter.com je Google Android v těsném závěsu za populárním operačním systémem iOS firmy Apple. V roce 2012 si prozatím udržuje Google Android druhou příčku před BlackBerry OS, SymbianOS a Windows Phone. Statistika použití je vyobrazena v následujícím grafu:



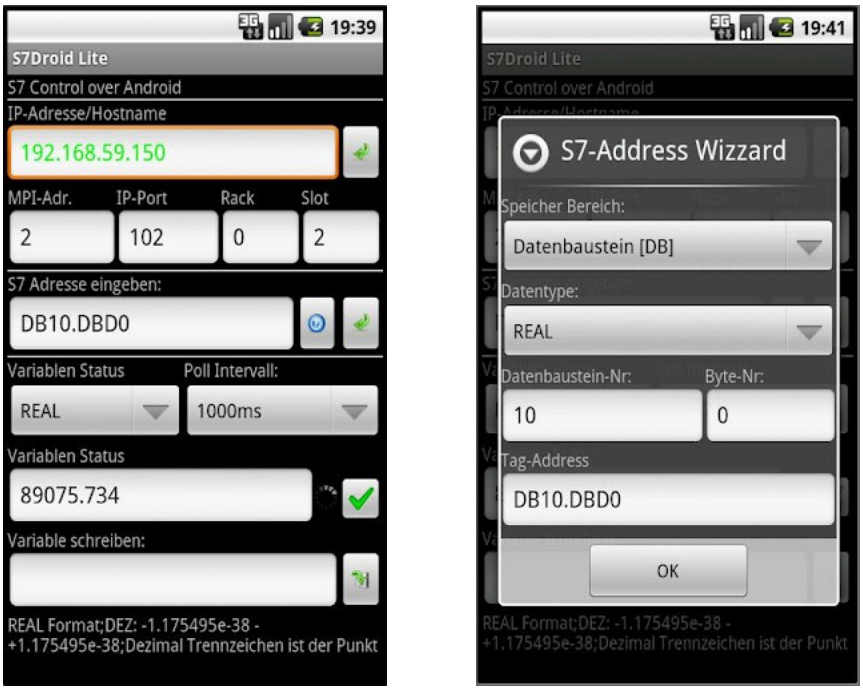
Obrázek 4.1 – Graf podílu mobilních operačních systémů na trhu za rok 2012

4.1 Současná řešení

V době zadání této diplomové práce nebyla na trhu žádná mobilní aplikace pracující výhradně s PLC SIMATIC. Avšak čerstvě od února roku 2012 již existuje jedno řešení a ještě k tomu právě pro operační systém Google Android. Tato aplikace nese název S7Droid a pro realizaci spojení s PLC využívá knihovnu libnodave. Vytvořila ji německá společnost Automation Software Engineering [6] a je dostupná ve dvou verzích:

- S7Droid Lite
- S7Droid Full

S7Droid Lite je zdarma ke stažení na Google Play (dříve Android Market) a obsahuje základní uživatelské rozhraní s možností čtení a zápisu hodnoty do vybrané oblasti paměti v PLC. V možnostech se dá vybrat PLC SIMATIC S7 řady 200, 300, 400, 1200 a Logo-0BA7. Zadáním IP adresy, cílového portu, čísla racku a slotu dojde ke spojení s PLC a odhalí se druhá polovina obrazovky, ve které se určí adresa hodnoty v paměti PLC, kterou má uživatel následně možnost monitorovat, případně měnit. Na obrázku 4.2 lze najít ukázky těchto obrazovek.



Obrázek 4.2 – Oficiální screenshoty aplikace S7Droid Lite ze stránek firmy Automation SE

Podle mého názoru je aplikace pro základní využití dostačující, ale pro nějaké profesionálnější využití bych ji nepoužil. S7Droid Lite je opravdu jen ukázková záležitost.

S7Droid Full je plnohodnotná placená varianta její Lite verze obsahující navíc i databázový systém, díky čemuž je uživatel schopen vytvářet organizované struktury technologického procesu, monitorovat je a celkově ovládat. Uživatelské rozhraní vypadá jednoduše a přesto účinně – uživatel má možnost přizpůsobit si obrazovku podle chuti. Přizpůsobit si může barvy tlačítek a textových popisků, počet záložek a další prvky. Aplikace je dostupná na Google Play a cenově vychází na 16,99€. Ukázky uživatelského rozhraní se nachází na obrázku *4.3*.

Obrázek 4.3 – Oficiální screenshoty aplikace S7Droid Full ze stránek firmy Automation SE

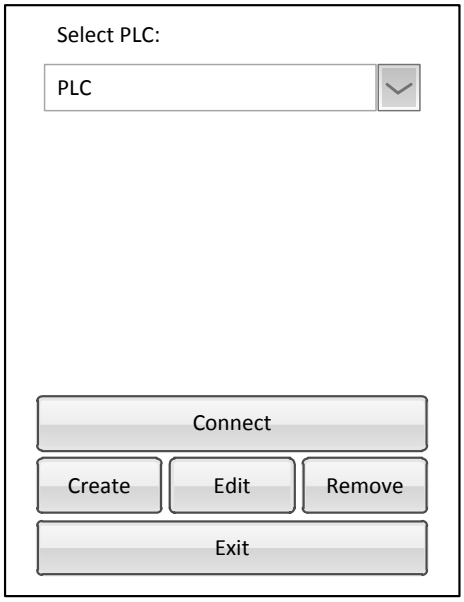
4.2 Analýza požadavků

Na základě požadavku firmy SIMPO CZ, s.r.o. by měly být vytvořeny dvě aplikace. První z nich má sloužit jako univerzální klient využitelný při práci v terénu, kdežto druhá z nich má sloužit jako monitorovací a zároveň ovládací aplikace napsaná přímo pro komunikaci s konkrétním PLC umístěném v sídle firmy. Obě aplikace by měly využívat pro spojení s PLC internetové připojení skrze Wi-Fi, nebo mobilní datové připojení. Při implementaci funkcí čtení a zápisu dat by se mělo dbát na omezení množství přenesených dat za účelem finančích úspor v případě využití mobilní datové sítě. Univerzální aplikace by měla mít možnost ukládat seznam monitorovaných PLC, a také data pointů pro každé uložené PLC.

4.3 Návrh aplikace Univerzální klient

4.3.1 Úvodní obrazovka

Tato aplikace by měla začínat obrazovkou, kterou bych nazval „rozcestníkem“, protože se na ní uživatel rozhoduje, na které PLC se hodlá připojit. Uživatelské rozhraní by mělo vypadat velice jednoduše. Návrh takové obrazovky je na obrázku *4.4*.



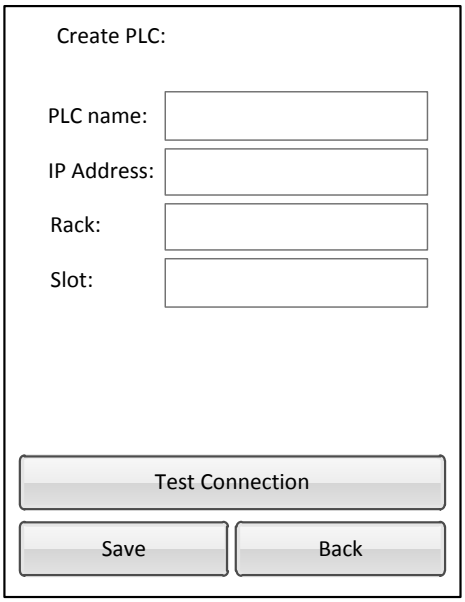
Obrázek 4.4 – Návrh úvodní obrazovky

Na úvodní obrazovce si uživatel bude moct vybrat ze seznamu řídicích systémů uložených v databázi aplikace. V případě, že v aplikaci ještě nebude uložený žádný záznam o PLC, použitím tlačítka Create se uživatel dostane do obrazovky pro vytvoření záznamu o PLC. Taková obrazovka by mohla vypadat jako na obrázku *4.5*.

Dalšími tlačítky majícími na starost práci se záznamem jsou tlačítka „Edit“ a „Remove“. Edit, neboli upravit, přesune uživatele do stejné obrazovky, jako tlačítko „Create“, akorát s tím rozdílem, že výsledkem uživatelovy činnosti na této obrazovce nebude vytvoření nového záznamu, nýbrž jeho editace.

Tlačítkem „Remove“ se aktuálně vybraný záznam o PLC smaže z databáze a pochopitelně tlačítkem „Exit“ se aplikace zavře.

4.3.2 Vytvoření nebo úprava záznamu o PLC

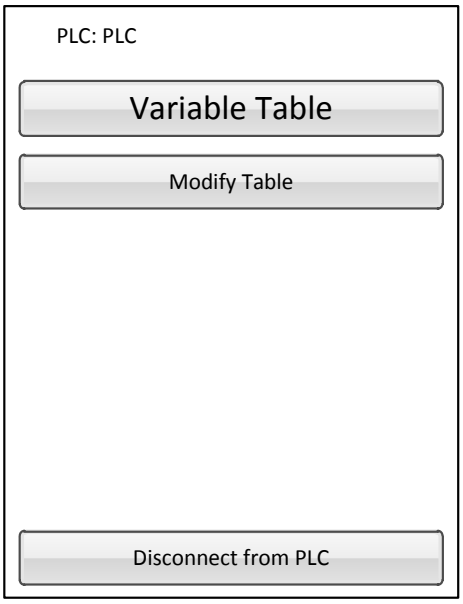


Obrázek 4.5 – Návrh obrazovky pro vytvoření záznamu o PLC

V případě tlačítka „Create“ se tedy uživatel přesune nové obrazovky, ve které by se mohla pro úspěšné uložení záznamu implementovat podmínka pro vyplněnost všech polí a následné otestování spojení. Při použití tlačítka „Test Connection“ pak dojde k testu a pokud by tento test dopadl úspěšně, povolilo by se použití tlačítka „Save“. Tlačítko „Back“ samozřejmě má za úkol návrat do předchozí obrazovky (čili do úvodního „rozcestníku“) bez vytvoření záznamu o PLC.

4.3.3 Křižovatka PLC

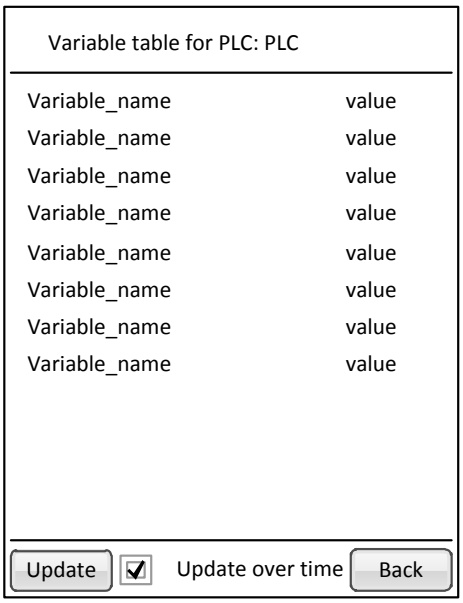
Když bychom se vrátili na úvodní obrazovku, zbývá ještě dodat funčknost tlačítka „Connect“. Pokud má uživatel vybrán řídicí systém, ke kterému se chce připojit, dotekem na tlačítko „Connect“ tak učiní a přesune se do další obrazovky. Tato nová obrazovka by mohla sloužit jako křižovatka pro dva další směry. Buď si uživatel bude chtít sestavit seznam monitorovaných hodnot, nebo tyto hodnoty monitorovat. Vzhled této obrazovky by neměl být nijak rozvitý, stačí pouze tři tlačítka znázorňující tři možné cesty. Dvě jsem již naznačil, třetí cesta je samozřejmě cesta zpět. Obrazovka by tedy vypadala jako na obrázku *4.6*.



Obrázek 4.6 – Návrh obrazovky křižovatky pro konkrétní PLC

4.3.4 Monitorování proměnných

Tlačítkem „Variable Table“ by si uživatel mohl v další obrazovce prohlédnout uložené záznamy o proměnných, které chce monitorovat. Tyto seznamy proměnných pro příslušné záznamy PLC v databázi aplikace se však musí nejprve vytvořit. Tvorba takových seznamů se provádí v obrazovce přístupné přes tlačítko „Modify Table“. Tlačítkem „Disconnect from PLC“ se uživatel samozřejmě vrátí na úvodní obrazovku s možností přepnutí na jiný uložený záznam o PLC. Návrh obrazovky zobrazené po použití tlačítka „Variable Table“ je naznačen na obrázku *4.7*.

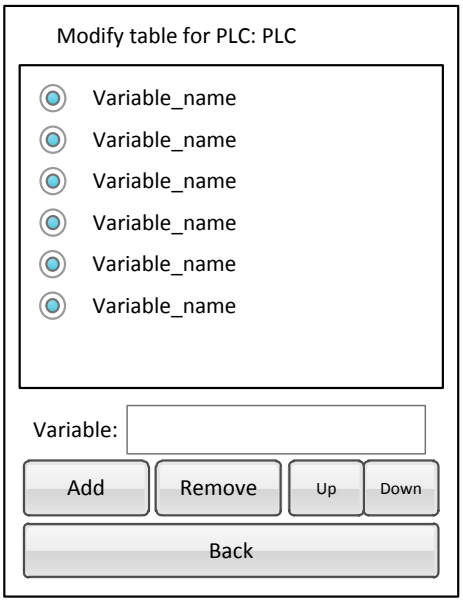


Obrázek 4.7 – Návrh obrazovky pro zobrazení monitorovaných hodnot

V hlavní části této obrazovky by se načetl seznam proměnných uložených v databázi. Tento seznam bude obsahovat název proměnné a její aktuální hodnotu. Ve spodní části obrazovky by se pak měla nacházet možnost aktualizace tohoto seznamu – hlavně tedy jeho hodnot. V případě zájmu o průběžnou aktualizaci může být vedle tlačítka aktualizace ještě i zaškrtávací políčko pro zapnutí automatické průběžné aktualizace těchto hodnot. Tlačítko „Back“ by pak vrátilo uživatele na křižovatku daného PLC.

4.3.5 Vytvoření seznamu monitorovaných proměnných

Pokud však uživatel bude chtít přednastavit seznam monitorovaných hodnot, měl by stisknout tlačítko „Modify Table“, které ho dostane do obrazovky s rozvržením jako na obrázku *4.8*.



Obrázek 4.8 – Návrh obrazovky pro sestavování seznamů monitorovaných hodnot

Na této obrazovce by měl mít uživatel možnost přidávat hodnoty pomocí zadání názvu proměnné (v PLC SIMATIC je to např.: *DB1.DBD0*), která se následně stisknutím tlačítka „Add“ přidá do seznamu ve formě přepínačů. Tlačítkem „Remove“ by se pak označená proměnná smazala ze seznamu, „Up“ by proměnnou posunula v seznamu o jednu pozici nahoru, „Down“ by posunula proměnnou v seznamu o jednu pozici dolů. Tlačítkem „Back“ by se samozřejmě obrazovka ukončila.

Co se týče univerzálního klienta, tento návrh by mohl pokrývat všechny požadavky zadavatele, tudíž se mohu pustit do práce na návrhu klienta „na míru“.

4.4 Návrh aplikace Klient „na míru“

Protože se bude jednat o aplikaci monitorující a ovládající konkrétní řídicí systém, je třeba pochopit technologii, kterou tento systém řídí. Pro snazší pochopení jednotlivých obrazovek této mobilní aplikace jsou v popisu technologie zahrnuty i návrhy určitých částí obrazovek, které by měly být později implementovány do aplikace. V této podkapitole popíšu nejprve návrhy dílčích části aplikace, ze kterých pak v podkapitole *4.4.6 Obecná struktura aplikace* sestavím návrh celkového rozložení aplikace.

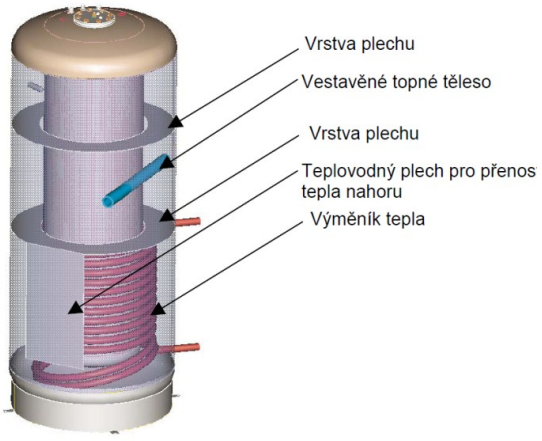
4.4.1 Popis technologie ohřevu topné a užitkové vody v obytném domě

Obytný dům se nachází v obci Třanovice, okr. Frýdek-Místek. Jedná se o dvougenerační dům, tedy má dvě bytové jednotky. Vytápění domu je realizováno prostřednictvím topné vody, v jedné bytové jednotce jsou využity k udržování teploty v místnostech radiátory, v druhé bytové jednotce slouží k tomuto účelu podlahové topení. Topná voda je ohřívána v plášťovém zásobníku, který má celkovou kapacitu 750l a ve kterém je současně ohřívána i užitková voda. Celý proces ohřevu a využití topné vody je řízen řídicím systémem Simatic S7 300 firmy Siemens.

Zásobník je tvořen dvěma válci:

1. Vnější válec o kapacitě 500l - jedná se vlastně o tepelný plášť sloužící k ohřevu
    a. užitkové vody
    b. topné vody pro radiátory v prvním podlaží
    c. topné vody pro podlahové topení v druhém podlaží
    d. topné vody pro ohřev bazénu
2. Vnitřní válec o kapacitě 250l – pro užitkovou vodu

Konstrukční uspořádání zásobníku je znázorněno na obrázku *4.9*.



Obrázek 4.9 – Schéma zásobníku vody s výměníkem tepla

Výměník tepla v dolní části boileru ohřívá topnou vodu solárními kolektory prostřednictvím nemrznoucí kapaliny, další zdroj – plynový kotel – ohřívá topnou vodu přímo. Také je možno využít elektrické topné těleso (spirálu), pro které je zde připravena jímka s utěsňovacím šroubem, nicméně v instalaci není tato možnost využita. Celková technologie je popsána schematicky na obrázku *4.10*.

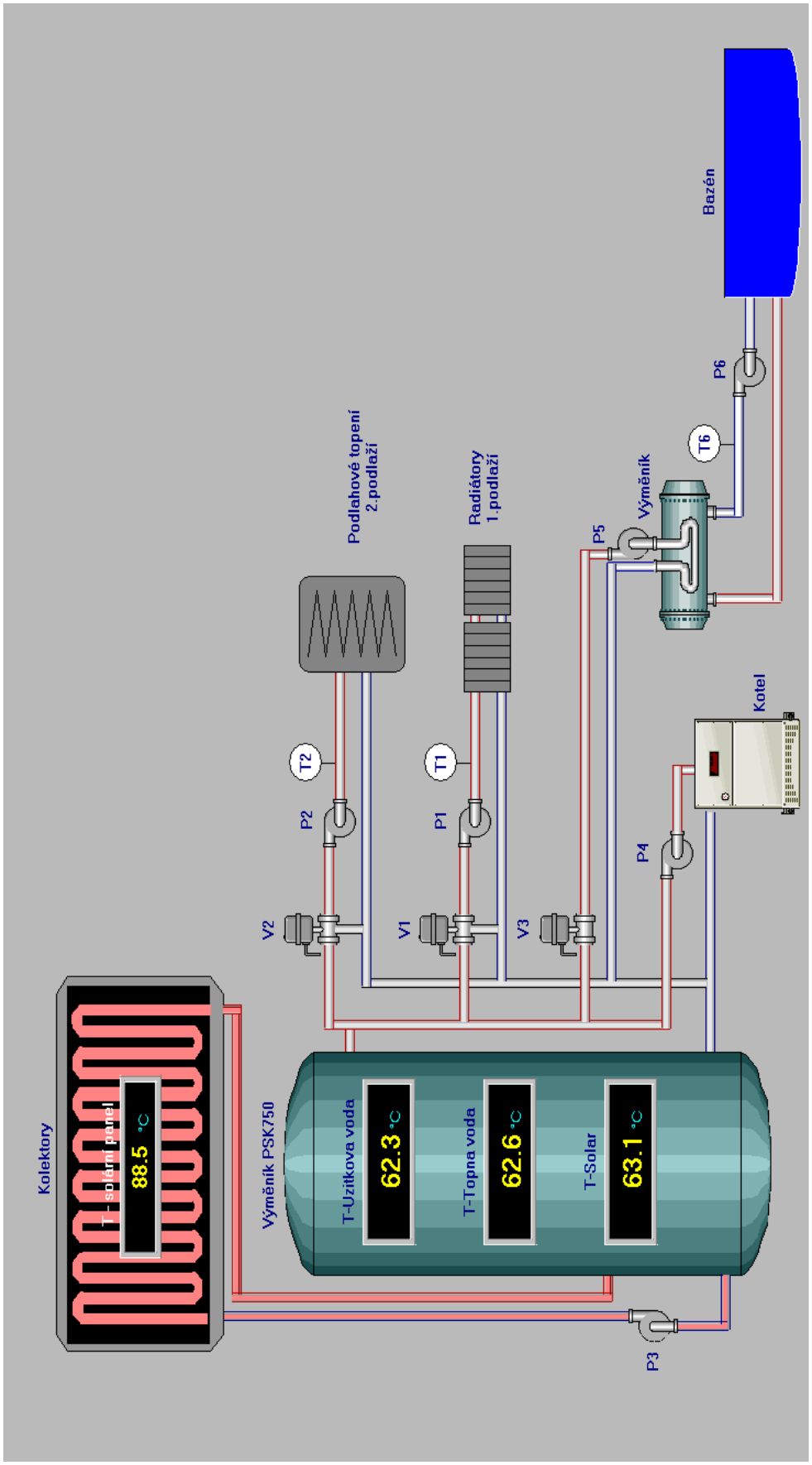
Jako zdroje tepla pro ohřev topné vody jsou zde použity:

- Plynový kotel
- Solární vakuové kolektory s vlastním nemrznoucím médiem

Spotřebiteli tepla jsou:

- Podlahové topení 2. podlaží
- Radiátorové topení 1. Podlaží
- Ohřev bazénu

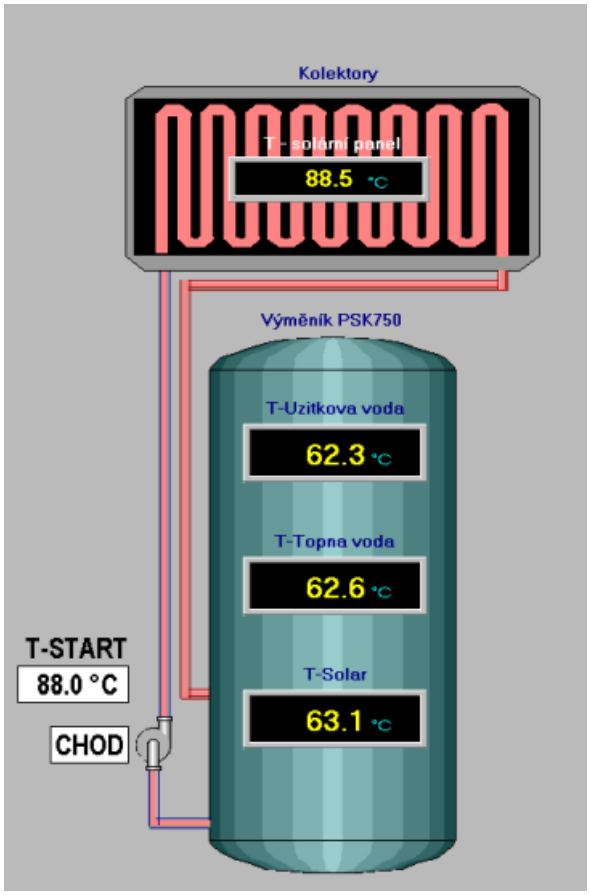
Systém ohřevu topné vody je celoroční. Principiálně pracuje tak, že za normálních podmínek (teplota topné i užitkové vody jsou nad min. limity) má maximální přednost ohřev pomocí solárních kolektorů. V případě dosažení stavu „přehřátého“ výměníku (v našem případě více než 65°C topné vody) dochází k odebrání nadbytečné energie tím, že se automaticky spustí ohřev bazénu. V případě, že bazénová voda již má 30°C, k tomuto ohřevu nedojde. Technické parametry této technologie (plocha kolektorů, kapacita topné vody, kapacita bazénu) ani příslušné klimatické podmínky neumožňují dosáhnout teploty vyšší než 75°C, takže do stavu, kdy by bylo možné dosáhnout teploty topné vody 100°C s nemožností „ochladit“ ji „přehřátým“ 30°C bazénem, se nelze prakticky dostat. Od dubna do září (za příznivého počasí) jsou hlavním zdrojem tepla solární kolektory, plynový kotel je zapínán vyjímečně.



Obrázek 4.10 – Celkové schéma technologie řízené PLC

4.4.2 Solární ohřev

Technologické schéma solárního ohřevu je zobrazeno na obrázku *4.11*.



Obrázek 4.11 – Schéma obrazovky solárního ohřevu

Systém vyhodnocuje rozdíl mezi teplotou média v solárních kolektorech a teplotou topné vody, která obklopuje solární výměník v dolní části výměníku PSK (T-Solar). V okamžiku dosažení nastavené diference mezi těmito teplotami je spuštěno čerpadlo, které vymění zahřáté médium ze solárních kolektorů s médiem ve výměníku. Impulsem pro start solárního čerpadla je tedy teplota

T-START = T-Solar + T-dif START

- T-START teplota média v solárních kolektorech, která startuje solární čerpadlo
- T-Solar teplota v dolní části výměníku
- T-dif START diference teploty kolektorů a T-Solar (zadávaný parametr)

Zadávání parametrů solárního ohřevu

Zadávání parametrů by se dalo provádět prostřednictvím ovládacího panelu podle návrhu na obrázku *4.12*. V okamžiku vyvolání tohoto panelu by mělo dojít k načtení aktuálních hodnot parametrů. Dotykem na příslušné zadávací okénko by se pak vyvolala klávesnice pro modifikaci příslušné hodnoty. Zadaná hodnota by se zapsala do řídicího systému tlačítkem „Uložit".

Obrázek 4.12 – Návrh ovládacího panelu pro zadání parametrů solárního ohřevu

Jednotlivé prvky ovládacího panelu a jejich významy:

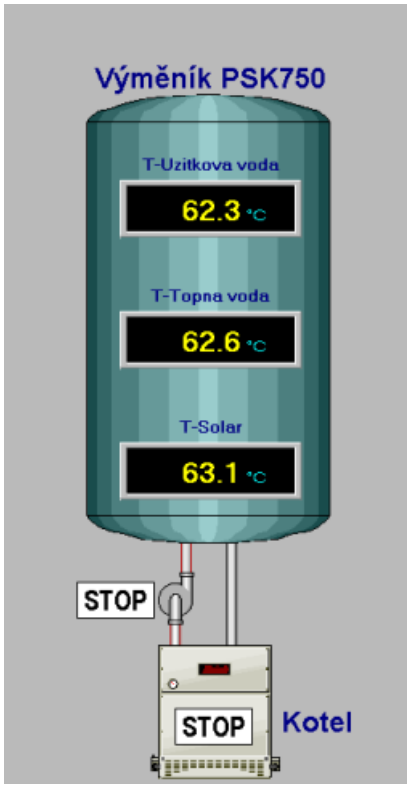
	Aktivní políčko pro zadání hodnoty
Uložit	Tlačítko pro uložení (zápis) hodnoty do řídicího systému
Získat aktuální hodnoty	Tlačítko pro načtení hodnot z řídicího systému
Zpět	Tlačítko pro zavření ovládacího panelu

Kromě základního parametru – diference teploty – jsou z důvodu ochrany proti zplynění média (nad 105°C) do systému ještě zavedeny další dva parametry.

T-DIF START	Rozdíl mezi teplotou média v solárních kolektorech a teplotou topné vody
T-MAX START	Maximální teplota v kolektorech pro start solárního řerpadla (ochrana proti zplynění média)
T-HAV STOP	Havarijní stop (blokáda) solárního čerpadla v případě, kdy už došlo ke zplynění média

4.4.3 Výměník PSK750

Jádrem celého systému je vlastní výměník. Je zdrojem teplé užitkové vody pro potřebu domácnosti a zdrojem tepla pro vytápění objektu. V zimním období a při neslunečném počasí je zdrojem tepla plynový kotel.



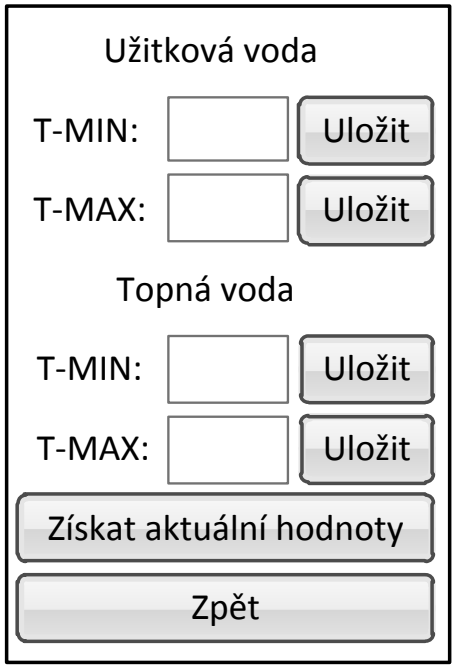
Obrázek 4.13 – Schéma obrazovky pro výměník PSK750

Řídicí systém má za úkol udržovat teplotu užitkové a topné vody v určitém rozsahu. Parametry řízení teploty topné a užitkové vody jsou následující:

- Užitková voda 45°C - 62°C
- Topná voda 40°C - 50°C

Zadávání parametrů užitkové a topné vody ve výměníku

Zadávání parametrů by se opět dalo prostřednictvím ovládacího panelu. Návrh zmíněného panelu je na obrázku *4.14*. V okamžiku vyvolání tohoto panelu by se načetly aktuální hodnoty parametrů. Dotykem na příslušné zadávací okénko by se vyvolala klávesnice pro modifikaci příslušné hodnoty. Zadaná hodnota se pak zapíše do řídícího systému tlačítkem „Uložit“.



Obrázek 4.14 – Návrh ovládacího panelu pro zadávání parametrů užitkové a topné vody ve výměníku

Jednotlivé prvky ovládacího panelu a jejich významy:

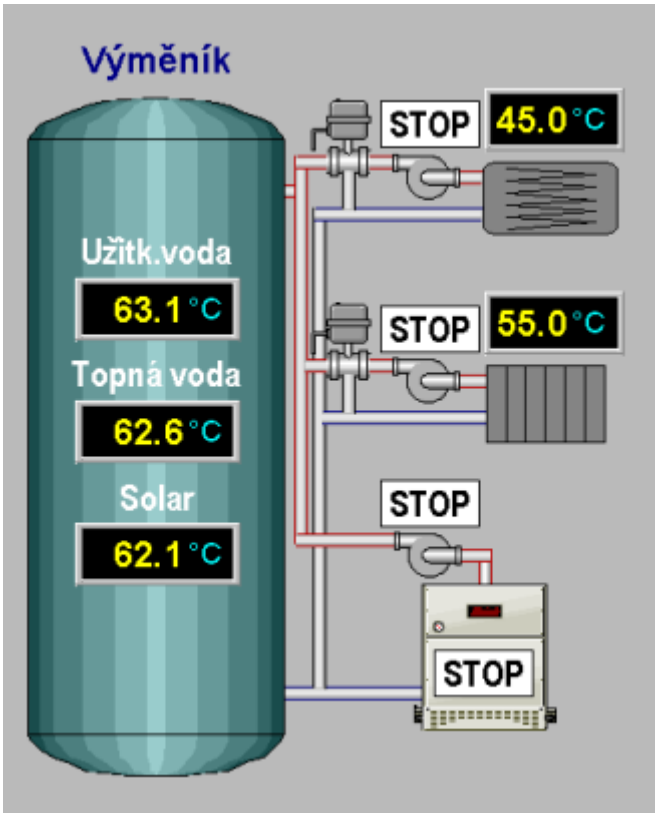
	Aktivní políčko pro zadání hodnoty
Uložit	Tlačítko pro uložení (zápis) hodnoty do řídicího systému
Získat aktuální hodnoty	Tlačítko pro načtení hodnot z řídicího systému
Zpět	Tlačítko pro zavření ovládacího panelu

Při hodnotě T-MIN kotel zapíná, při hodnotě T-MAX kotel vypíná.

Kotel se nezapne, pokud v posledních 30min nebyl aktivní solární ohřev. V případě, že teplota topné vody je nižší než 25°C, kotel se zapne vždy.

4.4.4 Topení

Topení je řešeno jednoduchým způsobem. Na základě požadavku (termostatu) je aktivováno vytápění v příslušné větvi. Nastartuje se čerpadlo v dané větvi a regulační ventil začne regulovat dle příslušné nastavené hodnoty. Pokud není teplota topné vody dostatečná aktivuje se ohřev topné vody plynovým kotlem.



Obrázek 4.15 – Schéma obrazovky pro topení

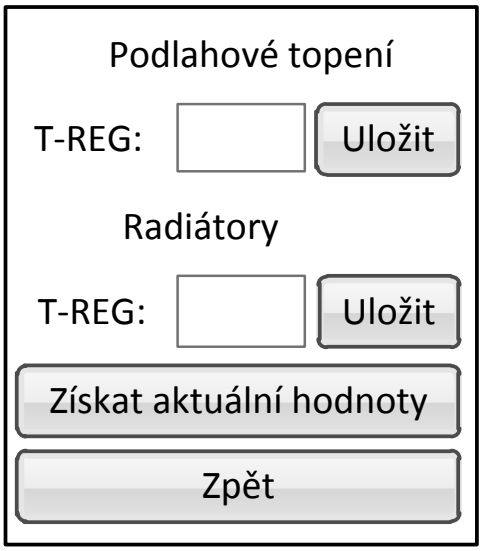
Řídící systém má za úkol udržovat teplotu topné vody v dané větvi, ve které je požadováno vytápění. Parametry řízení teploty topné a užitkové vody jsou následující:

- Podlahové topení 45°C
- Radiátory 50°C

Tyto parametry je možné zadávat a měnit prostřednictvím lokálního panelu nebo vzdáleného přístupu (mobilního zařízení).

Zadávání parametrů regulace topné vody při topení

Zadávání parametrů se může provádět prostřednictvím ovládacího panelu podle návrhu na obrázku *4.16*. V okamžiku vyvolání tohoto panelu se načtou aktuální hodnoty parametrů. Dotykem na příslušné zadávací okénko se vyvolá klávesnice pro modifikaci příslušné hodnoty. Zadaná hodnota se pak může zapsat do řídícího systému tlačítkem „Uložit".



Obrázek 4.16 – Návrh ovládacího panelu pro zadávání parametrů pro regulaci topné vody při topení

Jednotlivé prvky ovládacího panelu a jejich významy:

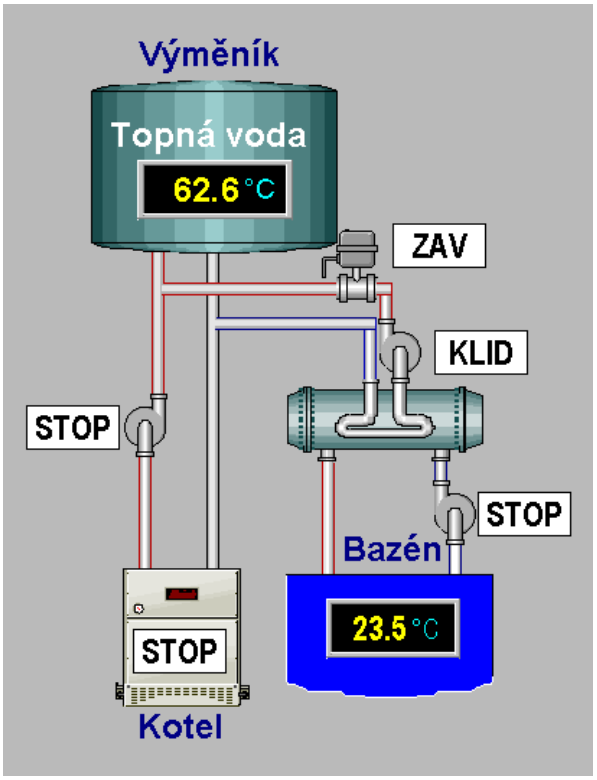
	Aktivní políčko pro zadání hodnoty
Uložit	Tlačítko pro uložení (zápis) hodnoty do řídícího systému
Získat aktuální hodnoty	Tlačítko pro načtení hodnot z řídícího systému
Zpět	Tlačítko pro zavření ovládacího panelu

Zadávané parametry **T-REG** nám určují, jakou teplotu topné vody chceme pro příslušnou topnou větev použít. Pro podlahové topení se nedoporučuje teplota vyšší než 45°C.

4.4.5 **Bazén**

Ohřev bazénu je realizován prostřednictvím malého výměníku, kde v primární části proudí topná voda a v sekundární části proudí bazénová voda. Maximální možná teplota bazénové vody je 30°C. Vlastní ohřev je prováděn dvěma způsoby:

1. Ohřev spouštíme vlastním povelem přes lokální panel nebo vzdálené připojení.
2. Ohřev je nastartován automaticky na základě tzv. „přebytku energie“, tj. když teplota topné vody přesáhne určitou nastavenou mez.



Obrázek 4.17 – Schéma obrazovky bazénu

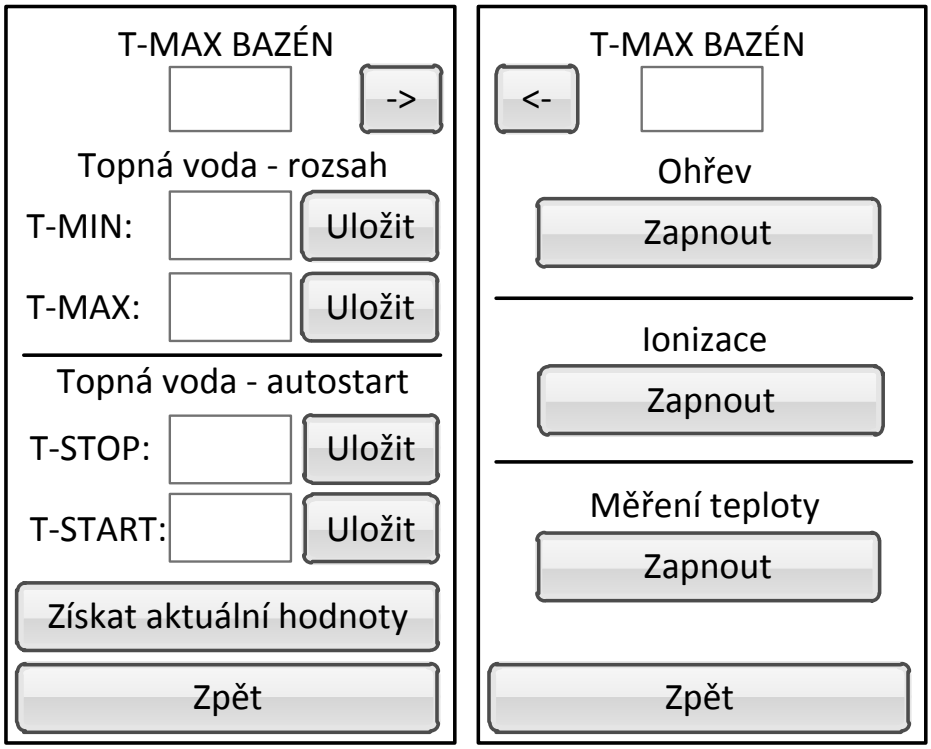
Z důvodu umístění měřicího teploměru blízko za malým výměníkem a ve velké vzdálenosti od bazénu, je nutné vždy před zahájením ohřevu změřit teplotu. Toto se provádí tak, že se zapne čerpadlo P6 na sekundární straně výměníku na dobu cca 5min za účelem cirkulace bazénové vody v měřicím potrubí. Po uplynutí doby měření (a v případě, že není teplota bazénové vody větší než 30°C) se otevře ventil V3 a spustí čerpadlo P5. Tím se dostane do primární části topná voda z velkého výměníku.

Základními nastavitelnými parametry tedy jsou:

- Max teplota bazénu 30°C
- Topná voda – použitelný rozsah 40°C – 70°C
- Topná voda – autostart 50°C – 65°C

Zadávání parametrů ohřevu bazénu

Zadávání parametrů se může uskutečnit prostřednictvím ovládacího panelu spuštěného v rámci mobilní aplikace. V okamžiku vyvolání tohoto panelu dojde k načtení aktuálních hodnot parametrů. Dotykem na příslušné zadávací okénko vyvoláme klávesnici pro modifikaci příslušné hodnoty. Zadanou hodnotu zapíšeme do řídícího systému tlačítkem „Uložit“.



Obrázek 4.18 – Návrh ovládacího panelu pro nastavení parametrů ohřevu bazénu

Jednotlivé prvky ovládacího panelu a jejich významy:

	Aktivní políčko pro zadání hodnoty
Uložit	Tlačítko pro uložení (zápis) hodnoty do řídícího systému
Získat aktuální hodnoty	Tlačítko pro načtení hodnot z řídícího systému
Zpět	Tlačítko pro zavření ovládacího panelu
-> <-	Tlačítka pro přepínání mezi panely

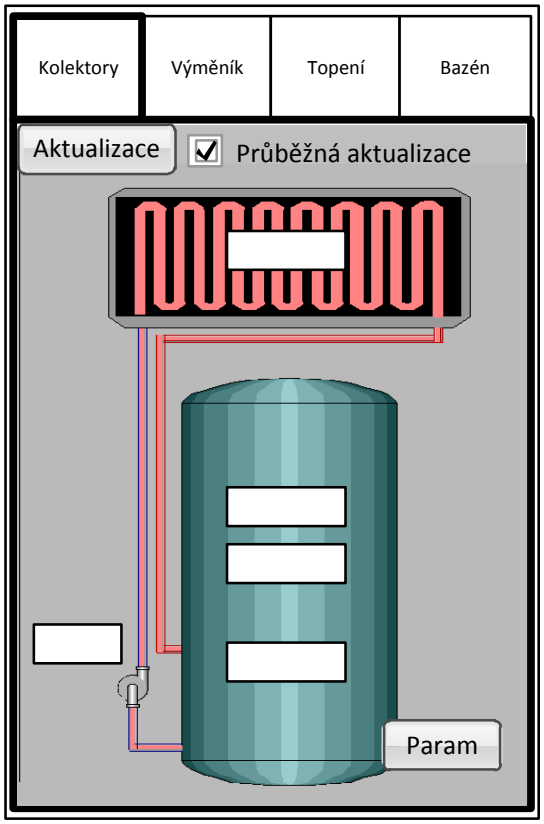
Zapnout	Tlačítko pro zapínání/vypínání akčního členu

Akčním členem se rozumí technologický prvek, jež se přímo ovládá technologickým povelem, ne pouze parametrizací, která v programu uvnitř řídícího systému na základě podmínky vyhodnotí technologický povel a ten pošle na akční člen.

Ohřev	Tento povel zapíná/vypíná čerpadlo bazénové vody
Ionizace	V sekundární části malého výměníku se nachází ionizační zařízení, které je spolu s čerpadlem bazénové vody zapnuto
Měření teploty	Zapnutí nebo vypnutí individuálního měření teploty bazénové vody. Tento povel pouze spustí na dobu cca 5 minut čerpadlo bazénové vody

4.4.6 Obecná struktura aplikace

Aplikace tedy musí pojmout celkem 4 části, které se dají logicky rozdělit podle toho, co se v dané části technologie ovládá. Protože je obrazovka mobilního telefonu relativně malá, je příhodné využít záložky, které pak zaručují snadné přepínání mezi jednotlivými částmi technologie. Rozvržení hlavní obrazovky by v nakonec mohlo vypadat jako na obrázku *4.19*.



Obrázek 4.19 – Návrh hlavní obrazovky

Bílá okénka představují pole, kde se bude zobrazovat příslušná hodnota podle toho, co se v té části monitoruje. Tlačítko „Param“ by pak spouštělo jednotlivé ovládací panely navržené v podkapitolách od 4.4.2 do 4.4.5. Spuštění takového panelu by ve skutečnosti mohlo vypadat jako na obrázku *4.20*.

Obrázek 4.20 – Návrh rozvržení obrazovky při spuštěném ovládacím panelu

5 Implementace

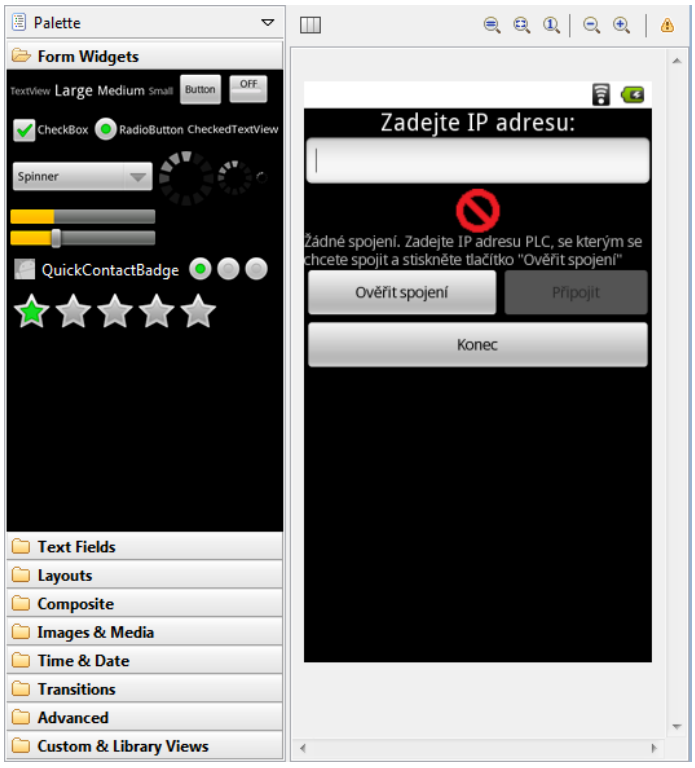
5.1 Google Android

5.1.1 Vývojové prostředí

Samotná implementace probíhala v Googlem oficiálně podporovaném vývojovém prostředí Eclipse, konkrétně ve verzi Indigo (3.7). Instalační balíky Eclipse v různých verzích lze najít na adrese http://www.eclipse.org/downloads/. Toto IDE je k dispozici zcela zdarma a bez nutnosti registrace. Velikost souboru činí zhruba 175 MB a stáhne se ve formě komprimované složky, takže instalace probíhá formou čisté extrakce.

Nezbytnou součástí pro vývoj aplikací na platformě Google Android je i Android SDK (Software Development Kit). Tento vývojový je k dispozici volně ke stažení na adrese http://developer.android.com/sdk/index.html. Tato instalace obsahuje i dvě další podstatné části:

- ADT plugin (Android Development Tool) - rozšíření pro Eclipse umožňující efektivní vývoj aplikací jednoduchou tvorbou uživatelských rozhraní pro Android. Ukázka tohoto pluginu je zobrazena na obrázku níže.



Obrázek 5.1 – ADT Plugin s vytvořeným uživatelským rozhraním

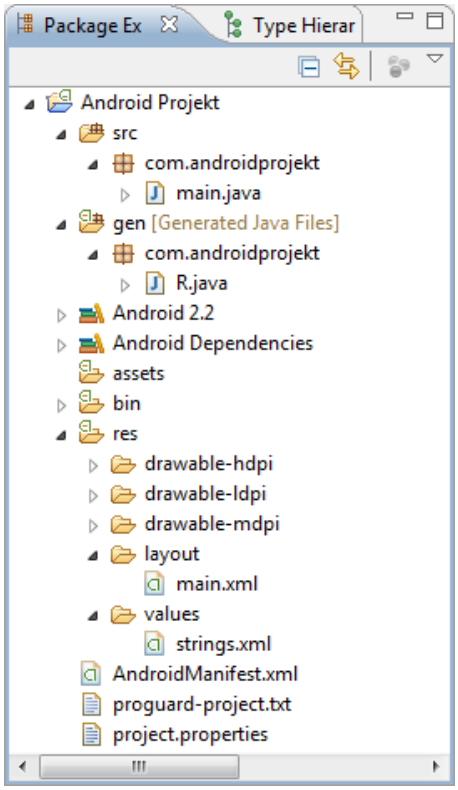
- AVD Manager - jednoduchý nástroj pro správu nastavení jednotlivých virtuálních zařízení emulovaných v Android Simulátoru, který je již také součástí instalace SDK.

5.1.2 Soubory v Android projektu

Aplikace pro Android jsou napsány v programovacím jazyce JAVA, takže se zde setkáváme se soubory s příponou *.java*. Zdrojové soubory pro rozvržení uživatelského rozhraní, soubory definující strukturu aplikace a soubory obsahující referenční proměnné pro propojení JAVY a prostředí Android aplikace jsou realizovány formou xml souborů.

Při založení nového projektu Android aplikace se automaticky vygeneruje několik souborů a složek. Pro snadnější orientaci v takovém projektu by bylo vhodné popsat význam těch zásadních.

Na obrázku *5.2* je zobrazena stromová struktura soborového systému projektu s názvem „Android Projekt“. V tomto kořenovém adresáři se nachází několik podložek.



Obrázek 5.2 – Stromová struktura projektu Android

- src – zdrojové soubory aplikace
- gen – automaticky generovaná referenční třída R.java propojující programovou část a grafickou část skrze unikátní identifikační jména všech objektů uložených ve složce „res“. Tato třída se při omylném smazání opět automaticky vygeneruje a neměla by se uživatelsky upravovat.
- res – složka obsahující soubory, jako jsou:
    - obrázky ve složkách „drawable“

  - uživatelská rozhraní ve formě xml souborů (složka „layout“)
  - seznamy použitých textových hodnot, stylů a formátování v souboru „strings.xml“ (složka „values“)
- AndroidManifest.xml – povinný soubor obsahující všechny základní informace o aplikaci, které předává operačnímu systému Android. Spouští se před všemi aplikačními zrdojovými kódy.

Příklad vygenerované třídy R.java může vypadat následovně:

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

5.1.3 Vývoj Android aplikací

Protože se v dalších podkapitolách věnuji samotnému programování, určitě se nevyhnu použití pojmů, které by si zasloužily na začátek trochu vysvětlit. Při vývoji aplikací pro Google Android se setkáváme se čtyřmi základními komponentami:

- Activity
- Intent Receiver
- Service
- Content Provider

Každá z těchto komponent splňuje jinou úlohu. Nejčastěji používanou komponentou a také základním stavebním kamenem pro aplikaci jako takovou je komponenta Activity.

Activity (Aktivita) reprezentuje většinou samostatnou obrazovku, která se v rámci aplikace může zobrazit. Android definuje v rámci aktivity několik předepsaných metod, které mají na starost různé úkoly. Příklady takových metod jsou: onCreate(), onPause(), onResume() a další. Není pravidlem, že Aktivita je pouze jediná obrazovka. Příkladem použití více aktivit na jediné obrazovce může být psaní sms zprávy. V jedné části obrazovky nalezneme seznam kontaktů, v druhé části samotný text zprávy. Každá tato část může být implementována jako dílčí aktivita. Při otevření nové Aktivity se ta předchozí zpravidla pozastaví (při pozastavení se volá právě metoda onPause()) a uloží se na začátek seznamu historie. V případě stisknutí tlačítka „Zpět“ se prakticky spustí poslední uložená Aktivita v seznamu historie.

Přepínání mezi obrazovkami se realizuje pomocí třídy Intent. Tato třída v českém překladu znamená „záměr“ a myslím si, že toto slovo samo o sobě popisuje smysl této třídy. Když bych to shrnul, tak třída Intent popisuje, jaký má aplikace záměr. Existují dvě základní části datové struktury třídy Intent: činnost (action), jaká se provede, a datový obsah, který se touto činností zpracuje. Typickou hodnotou pro činnost je MAIN, která funguje jako takové vstupní dveře do Aktivity, dalšími činnostmi jsou například VIEW, PICK, EDIT, atd.

Intent Receiver (Přijímač záměrů) je třída, která má na starost reagování aplikace na nějaké vnější události, jako je například příchozí hovor, nebo když je náhle dostupná bezdrátová síť, případně událost ve smyslu odbití dvanácté hodiny večerní apod. Tato třída může i nemusí zobrazit nějaké uživatelské rozhraní, které by dalo uživateli najevo, že se spustila daná událost. Při použití Intent Receiveru není nutné mít spuštěnou aplikaci. Pokud se spustí daná událost, systém to pozná a spustí aplikaci sám.

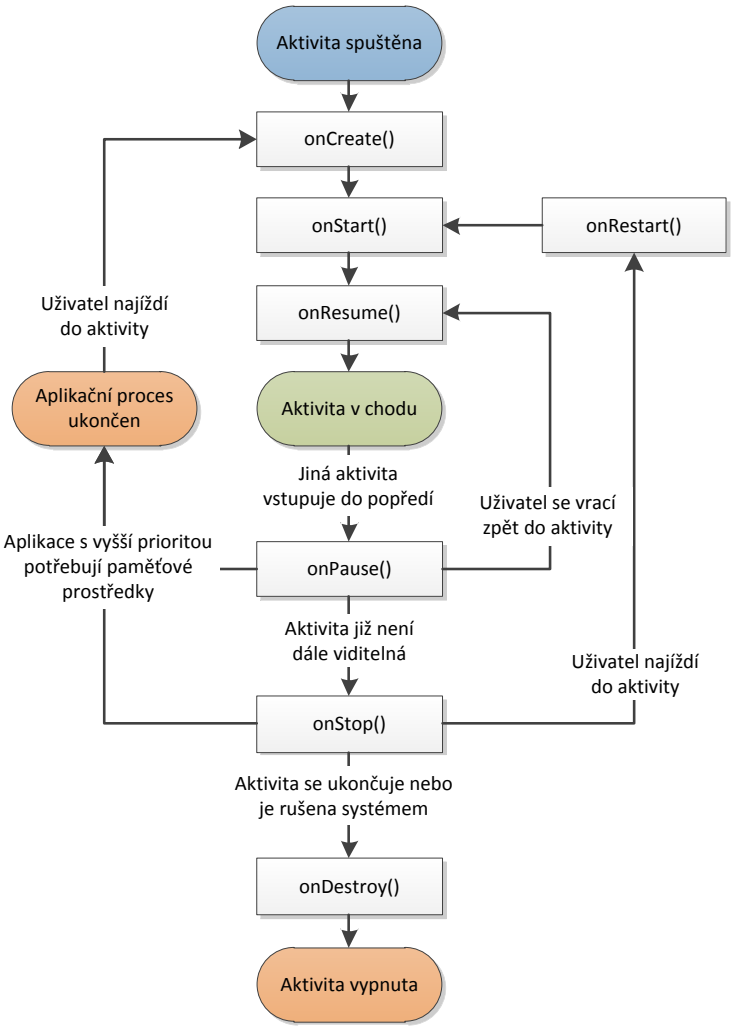
Service (Služba) je třída, která umožňuje spouštět určité požadované činnosti v pozadí bez uživatelského rozhraní. Velice dobrým příkladem je hudební přehrávač přehrávající písně ze seznamu skladeb. Služba funguje samostatně jako proces na pozadí a komunikuje s aplikací skrze rozhraní odhalené službou. Zařízení takto spuštěnou službu fungující jako hudební přehrávač udržuje v chodu, dokud neskončí přehrávání. Služba zůstává spuštěná i po vypnutí aplikace, tedy do doby, než splní to, na co byla naprogramována.

V některých aplikacích existuje potřeba ukládat data do nějakých datových struktur, ať už se jedná o soubory v paměti, nebo o SQLite databázi, pro takové případy existuje poslední důležitá komponenta s názvem Content Provider, která zprostředkovává možnosti tato data zpracovat.

Všechny další potřebné informace o vývoji aplikací na platformě Android jsou dostupné na [7].

5.1.4 Životní cyklus aktivity

Pro představu, jak se chová třída Activity v různých situacích, existuje v oficiální dokumentaci Google Android následující schéma:



Obrázek 5.3 – Schéma životního cyklu Aktivity

Aktivita má v základu čtyři stavy:

- Aktivní
  - V tomto stavu je aktivita v případě, kdy se nachází na popředí obrazovky a je aktivní.
- Pozastavená
  - Pokud se spustí další aktivita, která tu předešlou úplně nezastíní (tím je myšleno, že je transparentní, nebo není úplně překrývající předešlou aktivitu), zůstává předchozí aktivita ve stavu „Pozastavená“. Ponechává uložené informace o stavech jednotlivých členů aktivity v paměti, ale může hrozit vypnutí aktivity v případě nutnosti uvolnění paměti.
- Zastavená
  - Do tohoto stavu se aktivita může dostat ve chvíli, kdy se spustí nová aktivita a úplně překryje tu předcházející (čili není transparentní, nebo nemá menší

rozměry). Informace o stavech členů staré aktivity sice stále zůstávají uchovány v paměti, ale nejsou viditelné uživateli a daleko častěji hrozí možnost vypnutí staré aktivity v případě nutnosti uvolnění paměti.

- Neaktivní
    - V tomto případě je uvolněna paměť vynucením ukončení této aktivity, neuchovávají se žádné informace o stavech členů aktivity a v případě snahy uživatele vrátit se do této aktivity, aktivita musí být znovu spuštěna.

Po spuštění aplikace a načtení první aktivity musí proběhnout několik kroků, než se aktivita dostane do stavu „Aktivita v chodu“. Těmi kroky jsou metody, jejichž průběh se dá volitelně upravovat. Nejčastější metodou použitou i v mém vypracování je metoda onCreate(). Při vytváření aktivity je vhodné nastavit obsah příslušným uživatelským rozhraním. Toto rozhraní se dá vytvořit buď ručně, případně použitím ADT Pluginu, jenž byl zmíněn v kapitole 5.1.1. Výsledný xml soubor obsahující rozvržení jednotlivých prvků uživatelského rozhraní pak stačí použít v této metodě. Konkrétní příklad úpravy metody onCreate() si ukážeme v popisu implementace samotných klientů.

Metodu onStart() ve své aplikaci vůbec neupravuji, ale pro upřesnění tímto krokem aktivita prochází po proběhnutí metody onCreate(), čili při regulérním spouštění. Další možnou cestou, jak se dostat do tohoto kroku, je po restartu aktivity. Restartem je myšleno, že se aktivita dostala do stavu „Zastavená“ a musela projít metodou onRestart().

onResume() metodu používám v aplikaci častěji, protože pokud se aktivita dostane do stavu „Aplikace v chodu“, tak touto metodou prochází ve všech případech. Pokud chci, aby se mi pokaždé při najetí na danou aktivitu spustil určitý kód, s touto metodou mám jistotu.

Metoda onPause() je spuštěna vždy, když se aplikace zaměří na jinou aktivitu v popředí. Po této metodě se aktivita dostává do stavu „Pozastavená“.

onStop() metoda se spouští za podobných okolností, jako metoda onPause(), ale výsledkem je pak aktivita ve stavu „Zastavená“.

V případě úplného vypnutí aktivity zavoláním metody finish() nebo nečekaným ukončením procesu aplikace prochází aplikace před ukončením aktivity metodou onDestroy().

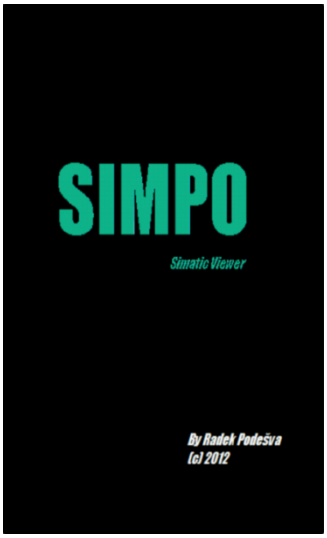
Pro základní orientaci ve vývoji aplikací na platformě Android by tento úvod mohl postačit, takže se nyní budeme zabývat samotnými aplikacemi.

5.2 Univerzální klient

Protože se návrh vcelku úspěšně ujal u zadavatele práce, nemusel jsem dělat mnoho změn, co se týče návrhu uživatelského rozhraní.

5.2.1 Uživatelské rozhraní

Spuštěním univerzálního klienta se nejprve otevře aktivita s logem firmy SIMPO a po tří-vteřinovém časovém intervalu se spustí hlavní aktivita. Třída *SplashActivity* v konečném důsledku vypadá jako na obrázku *5.4*.



Obrázek 5.4 – SplashActivity

Třída *SplashActivity* rozšiřuje třídu *Activity* z balíčku *android.app*. Při spouštění této třídy je nutné definovat rozvržení uživatelského rozhraní metodou *setContentView(layout)*. Parametr *layout* se většinou definuje jako veřejná statická proměnná třídy *R.java* zmíněné v podkapitole *5.1.2*. V případě *SplashActivity.java* jsem použil xml soubor s názvem „splash“, takže jako parametr stačilo dosadit *R.layout.splash*. Ve výsledku se načte vše, co je uloženo v souboru splash.xml.

Následný přesun do hlavní aktivity *StartActivity* se uskutečňuje pomocí třídy *Intent*, která je blíže popsána v kapitole *5.1.3*. Aktivita vytvoří samostatné vlákno nazvané „timer“, které se uspí na 3 vteřiny. Pokud nedojde k neočekávané vyjímce, použitím právě zmiňované třídy *Intent* se spustí další aktivita a *SplashActivity* se metodou *finish()* ukončí.

Po spuštění nové aktivity se stejným způsobem, jako u *SplashActivity,* opět nastaví rozvržení uživatelského rozhraní metodou *setContentView(R.layout.startactivity)*. Protože se však v souboru *startactivity.xml* nachází rozhodně více prvků, než v úvodní *splash.xml*, je potřeba tyto prvky inicializovat pro použití v třídě. Na obrázku 5.5 jdou tyto prvky jasně vidět.

a) *Bez záznamů o PLC* b) *Se záznamem o PLC*

Obrázek 5.5 – StartActivity

Ať už jde o *TextView*, *EditView* nebo *Button*, každý z těchto prvků patří do balíčku *android.widget*. Nastavení prvků probíhá voláním metody *findViewById(id)*, kde parametr „id“ představuje proměnnou použitou podobným způsobem, jako layout u *setContentView()* metody – odkazem na referenční třídu *R.java*. Ve výsledku pak například inicializace tlačítka „Create“ vypadá následovně:

```
Button btnCreate = (Button) findViewById(R.id.stac_btnCreate);
          // stac v mé aplikaci označuje prvek ze startactivity.xml
```

Jak je dále vidět na obrázcích *5.5a* a *b*, je tam patrný rozdíl. V prvním obrázku je možné stisknout pouze tlačítka „Create“ a „Exit“, kdežto ve druhém obrázku jsou všechna tlačítka stlačitelná. Navíc se v druhém obrázku nahradilo textové upozornění na chybějící záznam o uložených PLC z prvního obrázku za rozbalovací okénko, ve kterém už je vybráno PLC s označením „Tranovice PLC“. Stalo se tak z důvodu načtení seznamu uložených záznamů v PLC z databáze SQLite, která je

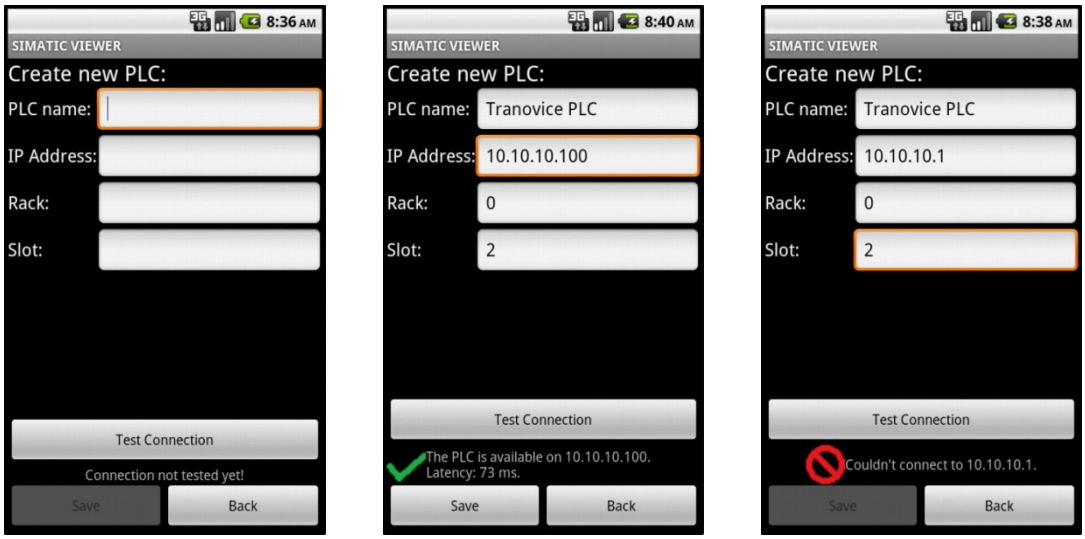
přímo podporovaná v rámci vývoje aplikací pro Android. Protože se s prací s databází ještě setkáme v mnoha případech, veškeré informace ohledně použití SQLite jsou sepsány v samostatné podkapitole *5.2.2 SQLite*. Načtení seznamu uložených PLC na obrázku *5.5b* proběhne jednorázově a je aktualizováno pokaždé, když aktivita prochází metodou *onCreate()*.

Tlačítka sama o sobě nefungují, pokud nenastavíme takzvané naslouchače. Pro tuto aplikaci jsem místo nastavování dílčích naslouchačů s vlastními metodami zajišťujícími akci po stisknutí prvku, implementoval rozhraní *OnClickListener* rovnou na celou třídu. Tímto jsem si zařídil přehledný kód a tím i jednodušší orientaci. V praxi pak vypadá deklarace třídy včetně nastavení naslouchače a zavedení metody *onClick(view)* následovně:

```
public class CreateEditPLCActivity extends Activity implements
OnClickListener {
	Button btnCreate = (Button) findViewById(R.id.stac_btnCreate);
	btnCreate.setOnClickListener(this);
	@Override
	private void onClick(View v){
		switch (v.getId()){
		case R.id.stac_btnCreate:
			// akce po stisknutí tlačítka
			break;
		}
	}
}
```

Po stisknutí tlačítka „Create“ se tedy spustí aktivita *CreateEditPLCActivity.java*, jejíž uživatelské rozhraní je vyobrazeno na obrázku *5.6a*. Tlačítko „Edit“ je přístupné ve chvíli, kdy se v rozbalovacím okénku v aktivitě *StartActivity* nachází alespoň jeden záznam o PLC. Pro rozlišení, jaké nastavení prvků má mít stejná aktivita spuštěná skrze jiná tlačítka se dá využít možnosti předávání parametrů pomocí třídy *Intent* a její metody *putExtra(key,value)*. Jako parametry se zadávají „key“ reprezentující jednoznačné označení obsahu předávaného parametru, a „value“ obsahující předávanou hodnotu. Ve spouštěné aktivitě se pak tyto předávané parametry přijímají pomocí instance třídy *Bundle* metodami začínajícími slovem „get“. Pro String hodnoty by to byla metoda *getString(key)*, kde parametr „key“ odkazuje na jednoznačné označení předávaného parametru ze spouštěcí aktivity.

V případě mé aplikace jsem se rozhodl použít globálních proměnných, které ukládám do třídy *Var.java*, které zahrnuji do rozlišovací podmínky v metodě *onCreate()* při spouštění *CreateEditPLCActivity* aktivity. Tento způsob není sice nejvhodnější pro mnoho aplikací, ale pro účely této aplikace bohatě dostačující.



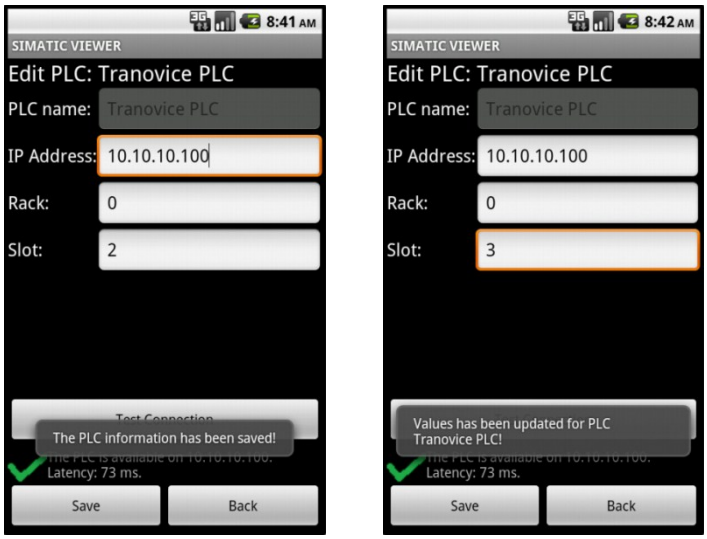
a) Čistý formulář *b) Úspěšný test spojení* *c) Neúspěšný test spojení*

Obrázek 5.6 - CreateEditPLCActivity

Pro uložení záznamu je nutné vyplnit všechna pole a úspěšně otestovat spojení s PLC. Testování spojení je realizováno třídou *NetworkCheck.java*. Tato třída obsahuje metodu *testConnection(address,port,timeOut)*, která na základě jednoduché realizace soketového spojení vrací hodnotu časové odezvy v milisekundách. Jako parametr *address* se v této metodě používá obsah pole „IP Address“ z formuláře, *port* je staticky nastaven na 102 (z důvodu použití ISO-TSAP protokolu pro realizaci spojení skrze knihovnu libnodave), a parametr *timeOut* označuje, jak dlouho má pokus o spojení probíhat, dokud test neskončí. V této aplikaci je výsledkem neúspěšného testu spojení znak červené stopky s příslušným textem, jak je vidět na obrázku *5.6c*. Po úspěšném spojení je uživatel informován zeleným háčkem a informací o odezvě (viz. Obrázek *5.6b*). Další podmínkou pro úspěšný test spojení je vyplněnost všech polí. Pole „IP Address“ navíc obsahuje textový filtr dovolující zápisu pouze znakového řetězce ve tvaru IP adresy. Nastavení filtru je řešeno v metodě *setFilterForIP()* ve třídě *CreateEditPLCActivity*.

Pokud test spojení tedy dopadne v pořádku, umožní se stisknutí tlačítka „Save“. Stisknutím tlačítka „Save“ pak proběhne zápis do SQLite databáze, která je rozepsaná v kapitole *5.2.2 SQLite*. Pokud je zápis do databáze úspěšný, proběhne celková změna této aktivity do podobné formy, jako kdyby uživatel záznam o PLC upravoval pomocí tlačítka „Edit“ nacházejícího se na úvodní aktivitě

StartActivity. Úspěšné uložení záznamu do databáze je doprovázeno malým textovým upozorněním realizovaným využitím třídy *Toast*. Vzhled *Toast* je vidět na obrázcích *5.7a* a *b*.



a) Uložení nového záznamu *b) Aktualizace stávajícího záznamu*

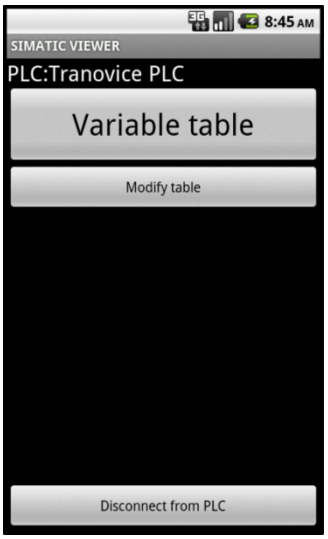
Obrázek 5.7 – Uložení záznamu do databáze a jeho aktualizace

Funkce tlačítka „Save“ při vytváření záznamu slouží jako příkaz *INSERT* v SQL, avšak druhotné použití tohoto tlačítka již funguje jako příkaz *UPDATE*. Po návratu do předchozí aktivity tlačítkem „Back“ již má uživatel vytvořen záznam o PLC, na které se může konečně úspěšně připojit pomocí tlačítka „Connect“ znázorněného na obrázku *5.5b*.

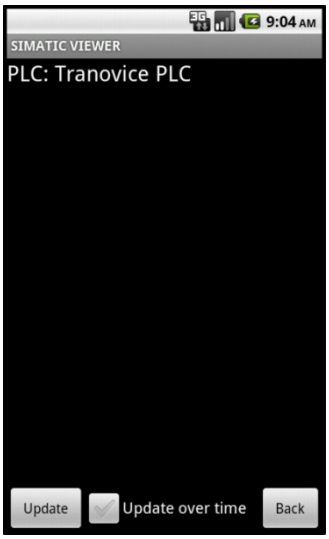
Pro spojení s PLC je do tlačítka „Connect“ na aktivitě *StartActivity* pro jistotu zapsána podmínka nového otestování spojení s uloženou IP adresou pro daný záznam o PLC. V případě úspěšného testu uživatel konečně spustí aktivitu *PLCMenuActivity*, která slouží v této aplikaci jako rozcestník mezi aktivitou pro vytváření seznamu monitorovaných proměnných, a aktivitou pro monitorování proměnných. Název PLC, který je zobrazen v horní části obrazovky je předáván opět přes globální parametr ve třídě *Var.java*. Vyobrazení této aktivity je na obrázku *5.8*.

Z aktivity PLCMenuActivity má uživatel možnost dostat se do dalších dvou aktivit. První z nich je *VariableTableActivity* pomocí tlačítka „Variable table“. Tato aktivita obsahuje seznam monitorovaných proměnných s aktuálními hodnotami. Aktivita vypadá jako na obrázku *5.9*. Po spuštění aktivity dojde k automatickému aktualizování hodnot v seznamu uložených proměnných. Tato akce proběhne jednorázově. Zároveň se spustí instance třídy ScheduledExecutorService, která zajistí pravidelnou aktualizaci seznamu proměnných a jejích hodnot. Tato instance plánovaně opakuje spouštění vlákna aktualizujícího texty proměnných i hodnot a tím zajišťuje pro uživatele vždy ty správné hodnoty, které se přečtou. Opakování je nastaveno na vteřinový interval. Tlačítkem „Update“ se provede opětovné čtení všech hodnot z PLC. Zaškrtnutím zaškrtávacího políčka „Update over time“

se v samostatném vláknu spustí opakované čtení hodnot z PLC. Tlačítkem „Back“ se uživatel samozřejmě vrátí na předchozí aktivitu.



Obrázek 5.8 – PLCMenuActivity



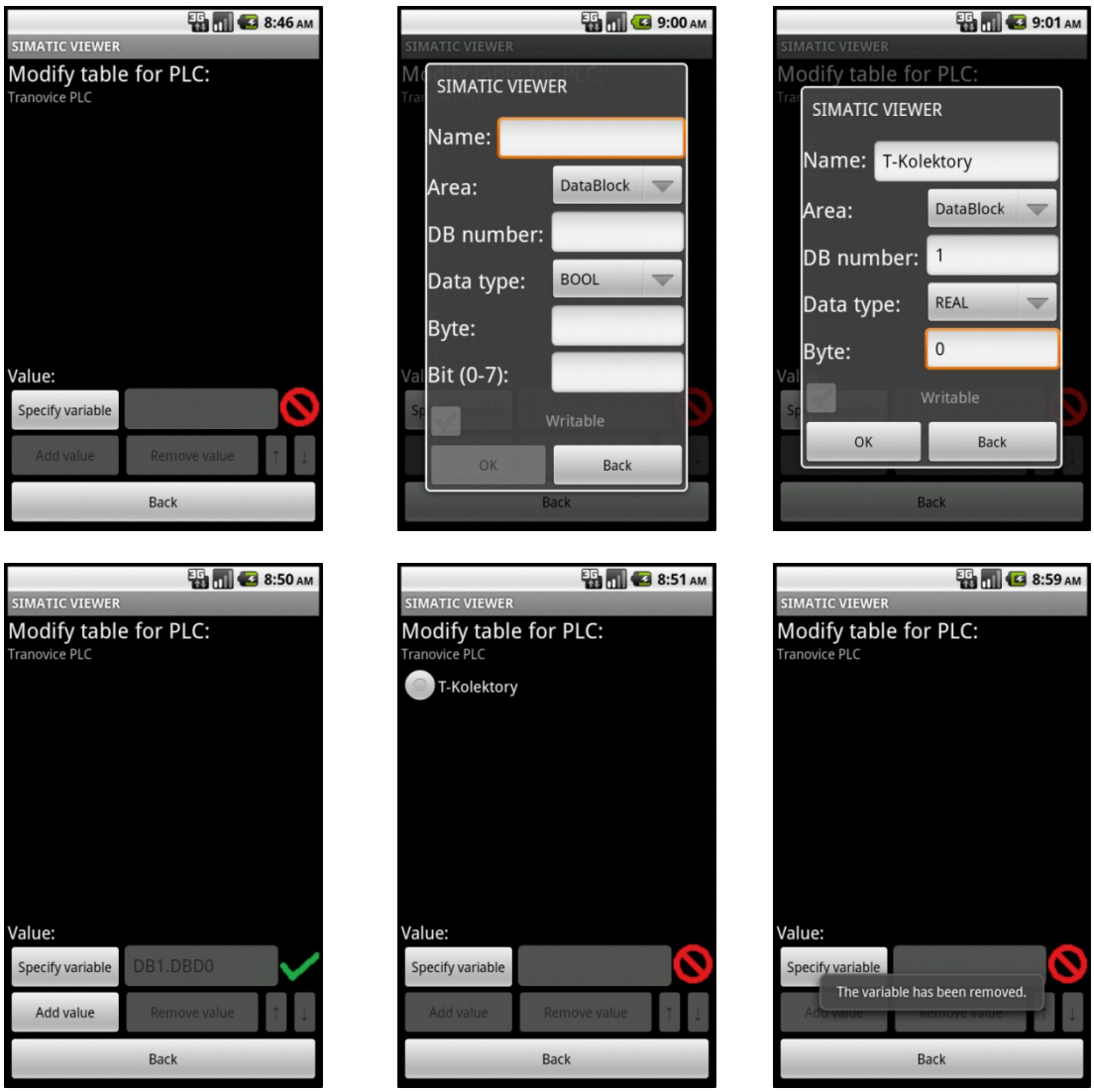
Obrázek 5.9 – VariableTableActivity

Zajímavější aktivitou je aktivita *ModifyTableActivity*, která slouží k vytváření seznamů proměnných, které se budou monitorovat. Po spuštění této aktivity proběhne načtení seznamu názvů proměnných pro dané PLC, které již byly v databázi uloženy. Vytvoří se dynamicky seznam přepínačů, na které se nastaví naslouchač *OnClickListener*. Aktivita pak vypadá jako na obrázku *5.10*.

Celý proces přidávání proměnných je realizován na základě zadání parametrů potřebných v metodě readBytes(...) třídy TCPConnection knihovny libnodave. Základní parametry jsou:

- Area – paměťová oblast
- DB number – v případě paměťové oblasti DataBloků se musí zadat i číslo DataBloku
- Data type – datový typ proměnné (BOOL, BYTE, REAL, INT, DINT, WORD, DWORD)
- Byte – startovní pozice bytu, z níž se má číst hodnota (délka čtení je určena datovým typem)
- Bit – v případě datového typu BOOL se musí zadat i číslo bitu

Celkový postup pro přidání proměnných je znázorněn na následující sérii obrázků:



Obrázek 5.10 – Postup přidávání proměnných

Pro přídání proměnné uživatel nejprve stiskne tlačítko „Specify variable“. Spustí se na popředí aktivity ModifyTableActivity nová aktivita, která v souboru *AndroidManifest.xml* využívá atribut *android:theme="@android:style/Theme.Dialog"*. Tato aktivita se jeví jako dialogové okno a slouží k nastavení parametrů proměnné. Jakmile uživatel vloží potřebné údaje a vyplní požadovaná pole, zpřístupní se tlačítko „OK“, které proměnnou nastaví do textového pole vedle tlačítka „Specify value“ ve formátu datového bodu, jaký je běžný pro PLC SIMATIC. Tento formát již byl zmíněn v návrhu aplikace v kapitole *4.3.5 Vytvoření seznamu monitorovaných proměnných*. Po stisknutí tlačítka „Add value“ se hodnota uloží do databáze a prostor uprostřed se vyplní aktuálním seznamem uložených proměnných.

5.2.2 SQLite

Aplikace používá k ukládání záznamů o PLC a také seznamů monitorovaných proměnných relační databázový systém SQLite. Tento systém je narozdíl od klasických databází založených na principu klient-server a spouštěných jako procesy pouze malou knihovnou, která se připojí k dané aplikaci a pomocí jednoduchého rozhraní ji využívá. Android používá tuto knihovnu, protože je velice malá a přesto efektní.

V mé aplikaci jsem vytvořil pro využití této knihovny třídu *MyPLCSQL*, ve které jsou definovány základní proměnné deklarující název databáze, názvy tabulek a názvy sloupců v jednotlivých tabulkách.

Pro vytvoření databáze a je v této třídě vytvořená podtřída *DbHelper*, která má za úkol vytvoření tabulek. Samotné vytvoření tabulek v databází se realizuje pomocí metody *execSQL(sqlquery)*, jejíž parametr „sqlquery“ funguje přesně tak, jak napovídá název. V aplikaci vytvářím celkem dvě tabulky. Tabulku „Servers“ a tabulku „Variables“. Jakmile jsou tabulky vytvořené, s databází se stále ještě nedá přímo pracovat. Pro práci s databází je neprve nutné otevřít spojení metodou *open()*. Pro ukončení spojení existuje metoda *close()*.

Ve třídě MyPLCSQL již zbývají vlastní metody pro práci s databází. Protože se z každé aktivity dotazuji na různé dotazy na databázi, je pro každou klíčovou funkci vytvořená vlastní metoda.

Například při spuštění aplikace Univerzální klient se vytvoří spojení s databází, proběhne metoda *getServers()* a ukončí se spojení. Metoda *getServers()* má návratovou hodnotu pole Stringů, které se využije jako parametr při vytváření obsahu rozbalovacího seznamu v úvodní obrazovce na obrázku *5.5b* v kapitole *5.2.1 Uživatelské rozhraní.*

Seznam metod ve třídě *MyPLCSQL* (výpis generován pomocí nástroje javadoc a upraven):

- createServer(server, ip, rack, slot)
  - vloží nový záznam o PLC do tabulky Servers
- createVariable(name, server, area, db, datatype, bajt, bit, writable)
  - vloží novou proměnnou do tabulky Variables
- deleteRow(server)
  - smaže záznam o PLC z tabulky Servers
- deleteRow(name, server)
  - smaže proměnnou z tabulky Variables
- getServerEdit(server)
  - načte hodnoty vztahující se k vybranému PLC

- getVariables(server)
    - načte seznam proměnných
- getVariablesForRead(server)
    - připraví dvoudimenzionální pole intů s hodnotami proměnných pro čtení z PLC
- switchRows(name, server, direction)
    - přehodí řádky se sousedním řádkem podle určeného směru (nahoru, dolů)
- updateServer(server,. ip, rack, slot)
    - aktualizuje záznam o PLC

5.2.3 Čtení hodnot z PLC

Čtení hodnot z PLC je realizováno pomocí nového vlákna vytvořeného třídou *readThread* (rozšiřující třídu *Thread*), která na základě String parametru „single“ nebo „overtime“ rozhoduje, zda bude probíhat jednorázově, nebo kontinuálně. Poté se hodnoty získané z databázové tabulky proměnných zabalí do dvoudimensionálního pole intů, které se pošlou jako parametr ke čtení do třídy *readFromPLC*. Ta již klasickým způsobem zprostředkuje komunikaci s PLC prostřednictvím knihovny libnodave. Po výměně informací vrátí třída readFromPLC, jako návratovou hodnotu obsahuje pole Stringů s proměnnými a jejich aktuálními hodnotami.

5.3 Klient „na míru“

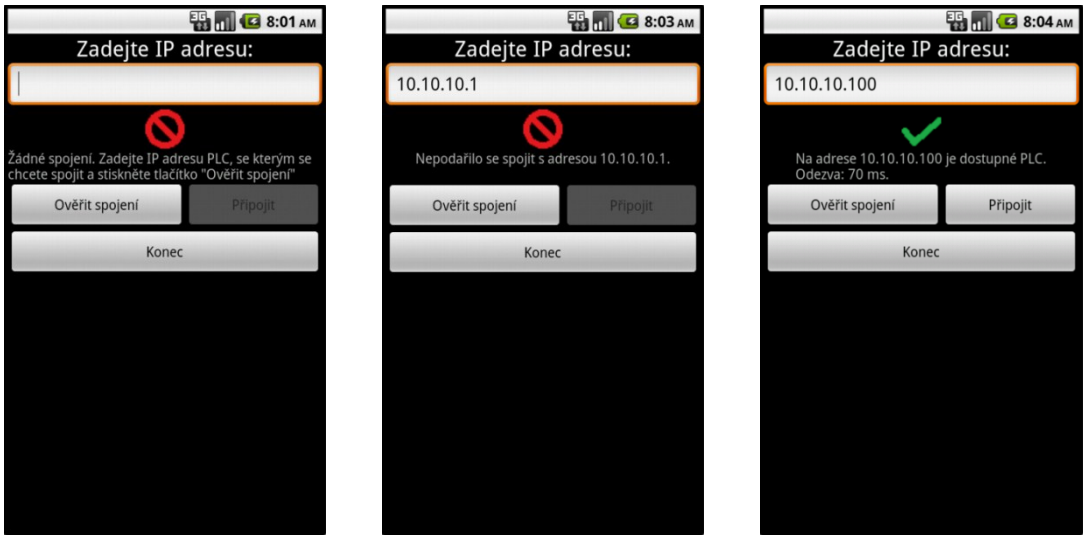
Protože klient „na míru“ slouží zpravidla ke stejnému účelu, jako univerzální klient z hlediska čtení hodnot, jejich implementace se principiálně moc od sebe neliší. Každá třída rozšiřuje třídu *Activity* a ty aktivity, ve kterých se nachází nějaké stisknutelné prvky, implementují ještě navíc rozhraní *OnClickListener*. V této aplikaci však přibyla ještě navíc možnost zápisu dat do PLC, protože u této konkrétní aplikace má uživatel jistotu, že nezpůsobí zápisem špatné hodnoty nežádoucí důsledek, protože zapisuje hodnoty do paměťových oblastí na adresy, které jsou vnitřně ošetřené podmínkami v samotné aplikaci PLC. Uživatel tak nemůže chybným zadáním povelů pro uzavření dvou třícestných ventilů vytvořit slepou uličku, do které by voda hnána čerpadlem mohla způsobit nadměrný tlak a hrozila by tak malá domácí katastrofa.

Samotná implementace této aplikace se tedy snaží odpovídat návrhu vytvořeném v kapitole *4.4 Návrh aplikace Klient „na míru“*.

5.3.1 Uživatelské rozhraní

Po spuštění aplikace má uživatel možnost zadat IP adresu, na které je tento konkrétní PLC dostupný. Existují minimálně dvě adresy, kterými může uživatel navázat spojení. První je vnitřní IP adresa sítě v domě, kde se PLC nachází, a druhá je vnější IP adresa směrovače od poskytovatele internetu. Směrovač je nastaven tak, aby přeposílal komunikaci podle určeného portu funkcí „Port Forwarding“. Protože komunikace knihovny libnodave vyžaduje port 102, poskytovatel připojení k internetu provedl na směrovači toto nastavení a PLC je tedy přístupné i zvenčí.

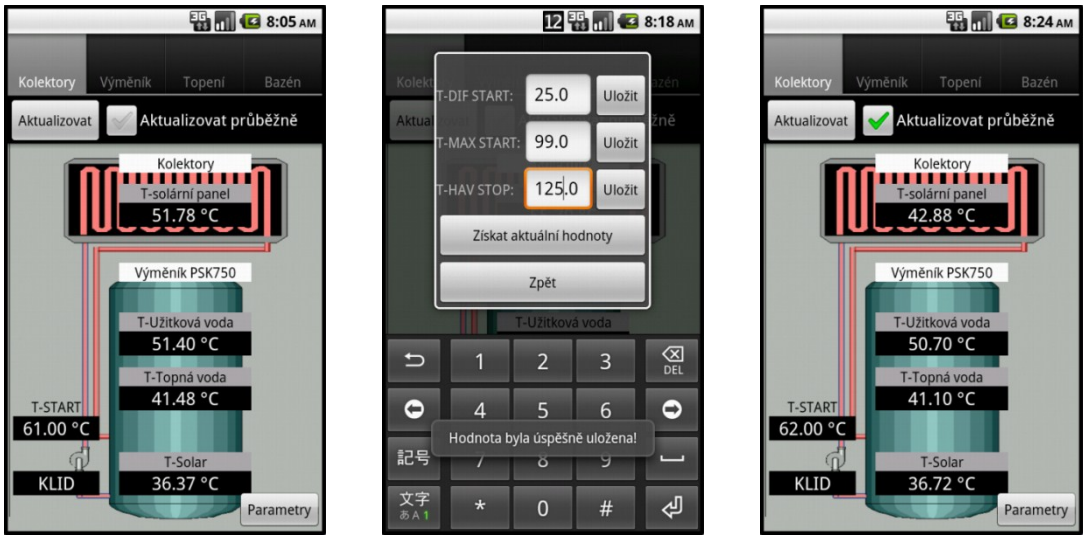
Úvodní obrazovka samotné aplikace je vyobrazena na obrázku *5.8*.



a) Bez zadání IP adresy *b) Špatně zadaná IP adresa* *c) Správně zadaná IP adresa*

Obrázek 5.11 – Úvodní obrazovka klienta „na míru“

Pole IP adresy má nastavený filtr, který byl také použit v aplikaci Univerzální klient, takže uživatel může zadat pouze IP adresu ve správné formě. Realizace spojení je samozřejmě podmíněna stiskem tlačítka „Ověřit spojení“. Pokud ke spojení dojde, zpřístupní se tlačítko „Připojit“, které již vede do hlavní aktivity, obsahující monitorované části technologie.



a) Monitorovaná technologie b) Zadávané parametry c) Zapnutá průběžná aktualizace

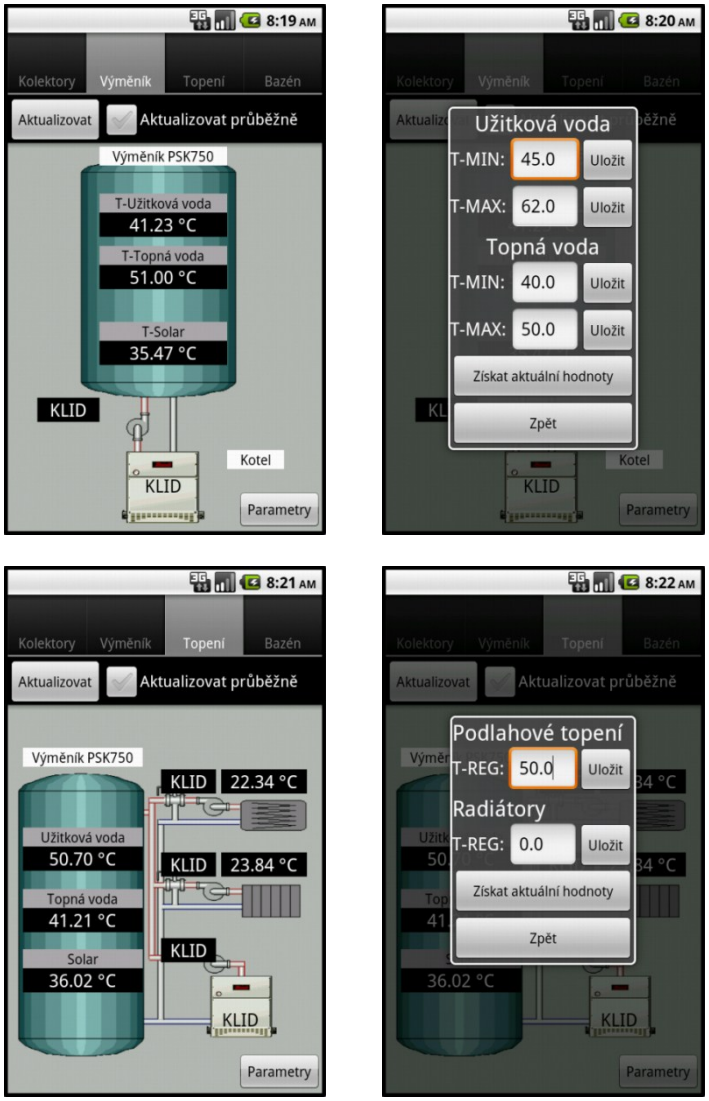
Obrázek 5.12 – První záložka obsahující část technologie nazvanou „Kolektory“

Jak je vidno na obrázku *5.9*, uživatelské rozhraní hlavní aktivity odpovída víceméně návrhu aplikace. Hlavní aktivita obsahuje čtyři záložky implementované třídami *TabHost* a *TabSpec*. Tyto záložky načítají samostatné třídy, ve kterých je definováno rozvržení uživatelského rozhraní podle toho, jak vypadá daná část technologie. Obrázek technologie je nastaven jako pozadí dané záložky a všechny textové prvky jsou pozicovány v tabulce tak, aby přibližně odpovídaly pozici dané monitorované části technologie.

Funkce tlačítka „Aktualizovat“ a zaškrtávacího políčka „Aktualizovat průběžně“ bude popsána v podkapitole *5.3.2 Čtení hodnot.*

Tlačítko „Parametry“ na každé záložce otevře aktivitu, která má v AndroidManifest.xml souboru specifikovaný atribut *android:theme="@android:style/Theme.Dialog"*. Tato specifikace způsobí otevření aktivity ve formě dialogu, takže ovládací panel pro zadávání parametrů ve skutečnosti vypadá jako malé dialogové okno.

Každá záložka tedy obsahuje podobné informace, takže nemá smysl je zbytečně zvlášť popisovat. Na následujících obrázcích jsou obrázky obrazovek z dalších záložek.



Obrázek 5.13 – Ukázky obrazovek dalších záložek s parametrizačními ovládacími panely

5.3.2 Čtení hodnot

Po stisknutí tlačítka „Aktualizovat“ proběhne čtení hodnot využitím stejného principu založení vlákna pro čtení, jako u Univerzálního klienta. Jediný rozdíl je v sestavení parametrů pro metodu *readBytes(...)* třídy *readFromPLC*, protože v této aplikaci se uživatel dotazuje pokaždé na stejné adresy, takže má možnost přečíst jedním zavoláním metody *readBytes(...)* větší rozsah bytů, a tím snížit časovou náročnost aplikace na získání aktuálních hodnot.

Příklad čtení:

```
dc.readBytes(Nodave.DB, 1, 0, 80, null);
Var.T_Kolektory = dc.getFloat();
Var.T_PSK_Solar = dc.getFloat(38);
Var.T_PSK_TopnaVoda = dc.getFloat(76);
```

Jak je vidět v kódu, je přečten rámec 80 bytů, z čehož se uloží do proměnných pouze čtyřbytové hodnoty. Maximální velikost rámce, který jsem byl schopen přečíst v jednom volání metody, bylo 102 bytů.

Abych se vyhnul čtení všech hodnot najednou, což by vedlo k dlouhému čekání na hodnoty, rozhodl jsem se podmínkou omezit počet čtených hodnot. Podmínka je závislá na aktuální záložce, která je zvolená. Při změně záložky se změní i číslo v globální proměnné *Var.Tab*, a tím i seznam hodnot, které se mají v danou chvíli číst. Tato podmínka je realizována i pro ovládací panely jednotlivých částí technologie.

5.3.3 Zápis hodnot

Zápis hodnot je implementován samostatným vláknem podobným, jako je vlákno pro čtení hodnot. Protože aplikace pracuje pouze s proměnnými v paměťové oblasti DataBloku, posílají se do metody *writeBytes(...)* pouze parametry čísla databloku a počáteční byte, kde se daná proměnná nachází.

6 Závěr

V rámci této diplomové práce jsem ve spolupráci s firmou SIMPO CZ, s.r.o. řešil problematiku spojenou s monitorováním technologie pro regulaci a ohřev vody v rodinném domě. Mým úkolem bylo prostudovat komunikační možnosti mezi konkrétním řídicím systémem SIMATIC S7-300 a mobilním zařízením a najít vhodné řešení pro implementaci aplikace pro mobilní zařízení schopné monitorovat a ovládat PLC prostřednitvím bezdrátového internetu, případně použitím mobilní datové sítě.

Vhodným kandidátem pro realizaci takové mobilní aplikace se jevila komunikační knihovna libnodave. Po nastudování možností implementace této knihovny do projektu mobilní aplikace jsem knihovnu použil a výsledek byl uspokojující. Tímto se mohlo se přejít k samotné části analýzy, návrhu a implementace mobilní aplikace.

Jako platforma pro vývoj mobilní aplikace byla zvolena na základě požadavků zadavatele platforma Google Android. Během analýzy si zadavatel ujasnil, že by výsledkem měly být dvě aplikace. První z nich měla být univerzální klient, který by měl možnost připojení na jakýkoli řídicí systém SIMATIC řady S7-300 s modulem komunikačního procesoru CP 343-1 připojeným k internetu. Druhá z aplikací měla být přizpůsobena konkrétní technologii provozovanou v rodinném domě. Obě aplikace se mi podařilo vytvořit a tím splnit zadání zadavatele.

Na aplikacích by se podle mého názoru ještě mohlo v budoucnu zapracovat, protože v ní chybí řešení zabezpečení. Věřím, že i otázka optimalizace by mohla aplikaci výrazně pomoct.

Největším přínosem této diplomové práce byla demonstrace možností při využití knihovny libnodave, která má v průmyslu tam, kde se pro automatizované řízení používá PLC SIMATIC, obrovský potenciál.

Použitá literatura

1. Průmyslový Ethernet II: Referenční model ISO/OSI. [online]. [cit. 2012-04-22]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=34209
2. GNU Lesser General Public License. [online]. [cit. 2012-04-10]. Dostupné z: http://www.gnu.org/copyleft/lesser.html
3. Hergenhahn, Thomas. LIBNODAVE, a free communication library for Simatic S7 PLCs. , LIBNODAVE -- Exchange data with Siemens PLCs. [Online] 10. 5 2010. http://libnodave.sourceforge.net/
4. Siemens AG, SINEC H1 Communication Processor User Manual [online]. 10/94, [cit. 2012-04-23]. Second edition http://cache.automation.siemens.com/dnl/jM2MDE3AAAA_1935233_HB/Q2802340.pdf
5. HEROUT, Pavel. *Učebnice jazyka Java*. 1. vyd. České Budějovice: Kopp, 2001, 349 s. ISBN 80-723-2115-3.
6. S7Droid. [online]. [cit. 2012-04-29]. Dostupné z: http://www.automation-se.de/
7. Android Development. [online]. [cit. 2012-05-03]. Dostupné z: http://developer.android.com/index.html
8. CP343-1. [online]. [cit. 2012-05-09]. Dostupné z: http://cache.automation.siemens.com/dnl/zY/zYzMDc1AAAA_8777308_HB/ghb3431_e.pdf